# Machine learning in computer vision

Lesson 9

# When to stop learning?

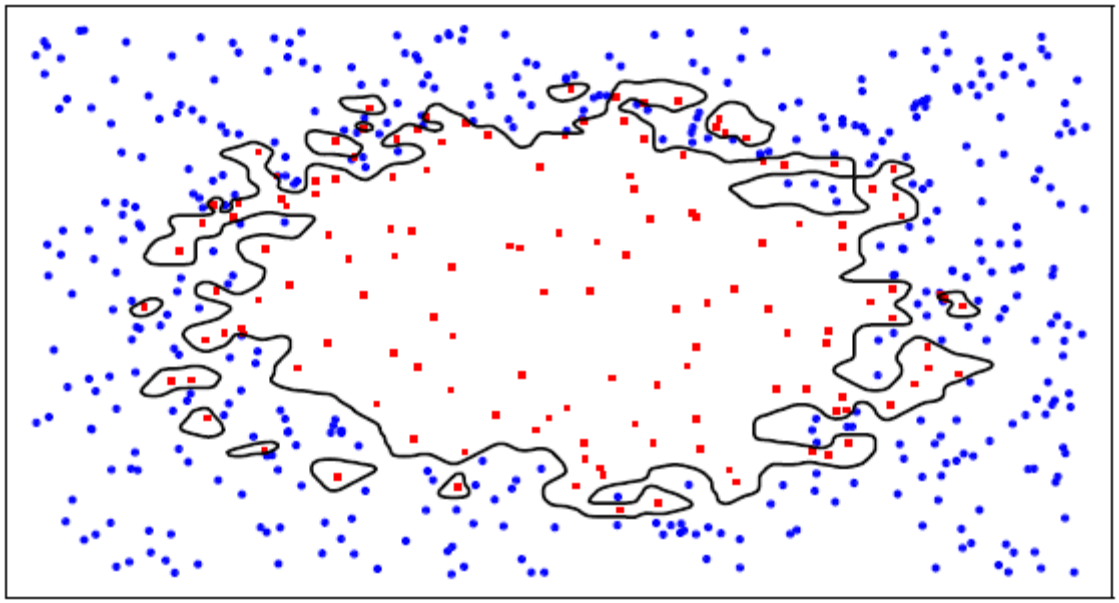
Max iteration

Goal achieved

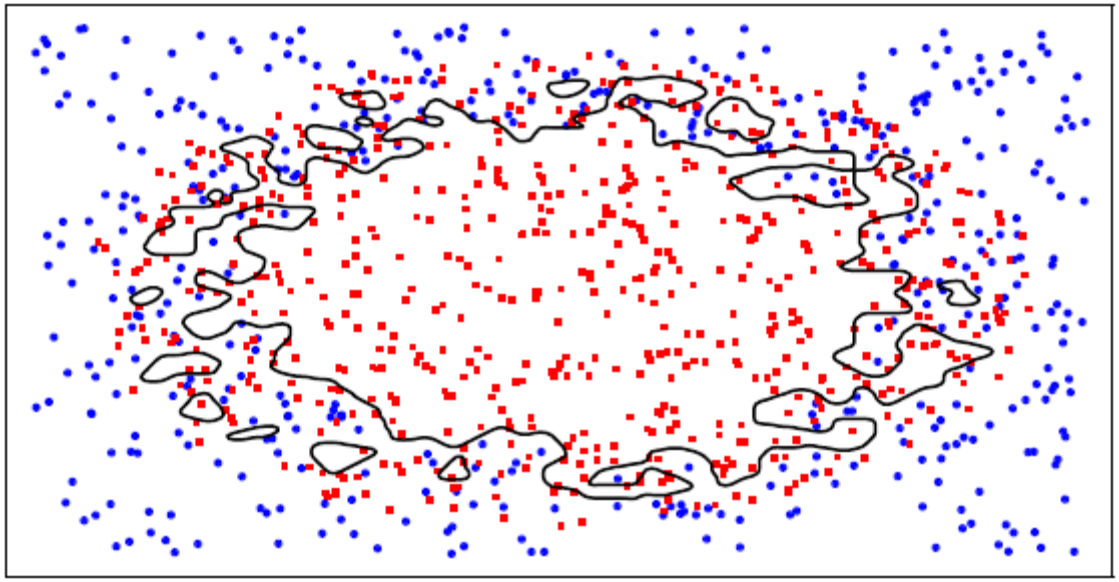
We arrive at the global minimum of the error function

Problem

- Overfitting – small training error, big testing error



Training set

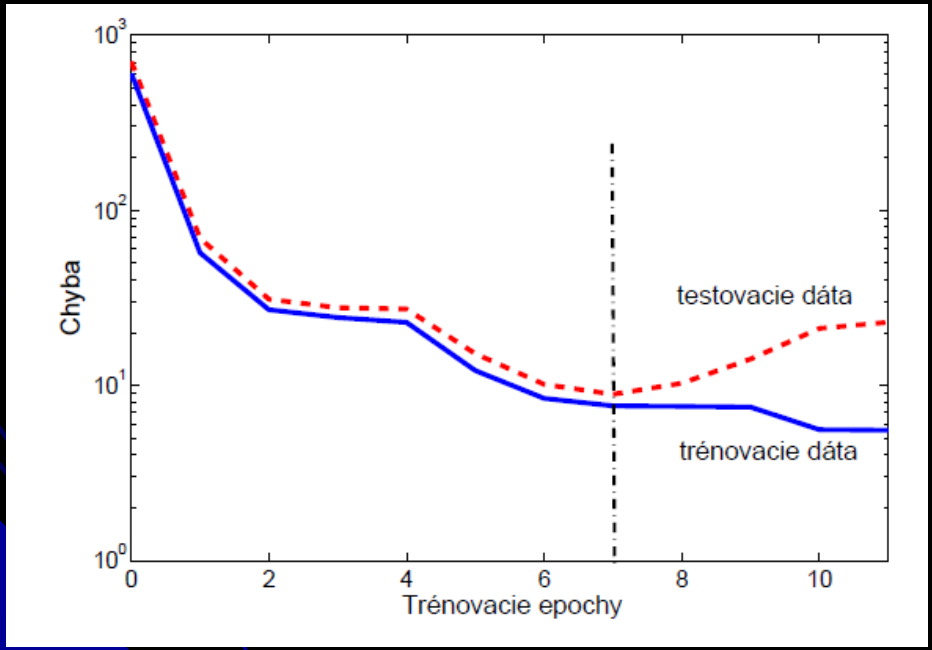


Test set

# Early stopping

Train and validation sets

stop when the validation error starts to grow



# Number of Hidden Layers

| # | Result |
|---|---|
| none | Only capable of representing linear separable functions or decisions. |
| 1 | Can approximate any function that contains a continuous mapping from one finite space to another. |
| 2 | Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy. |

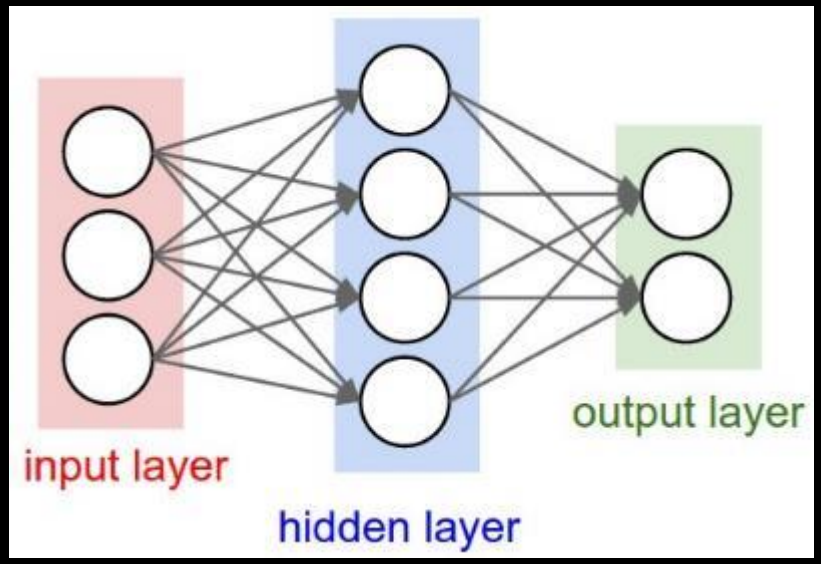
Jeff Heaton. 2008. Introduction to Neural Networks for Java, 2nd Edition (2nd ed.). Heaton Research, Inc.

# Number of neurons in the hidden layers

rules-of-thumb:

- between the size of the input layer and the size of the output layer.
- 2/3 the size of the input layer, plus the size of the output layer.
- less than twice the size of the input layer.

# Number of parameters



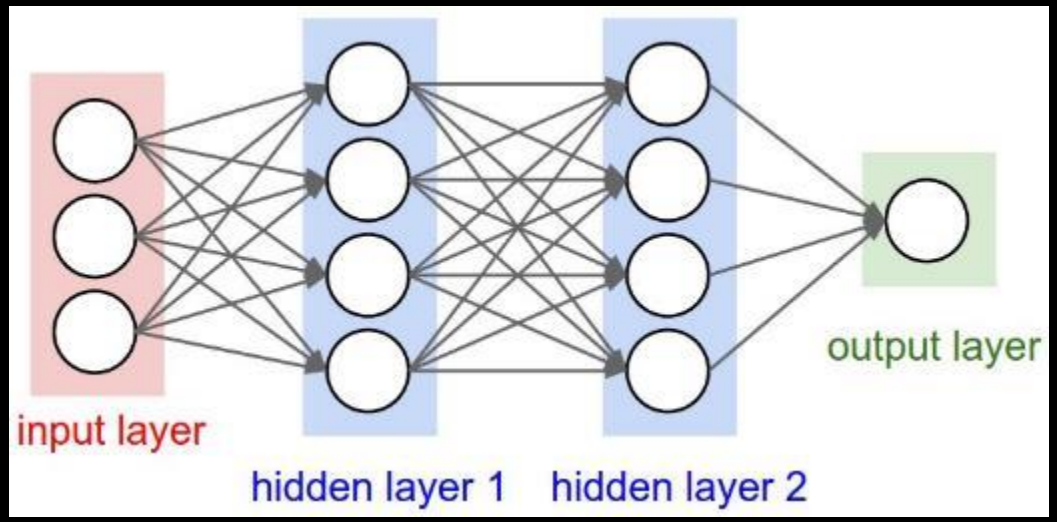

Weights and biases:

[3 x 4] + [4 x 2] = 20 weights

4 + 2 = 6 biases

http://cs231n.github.io/neural-networks-1/

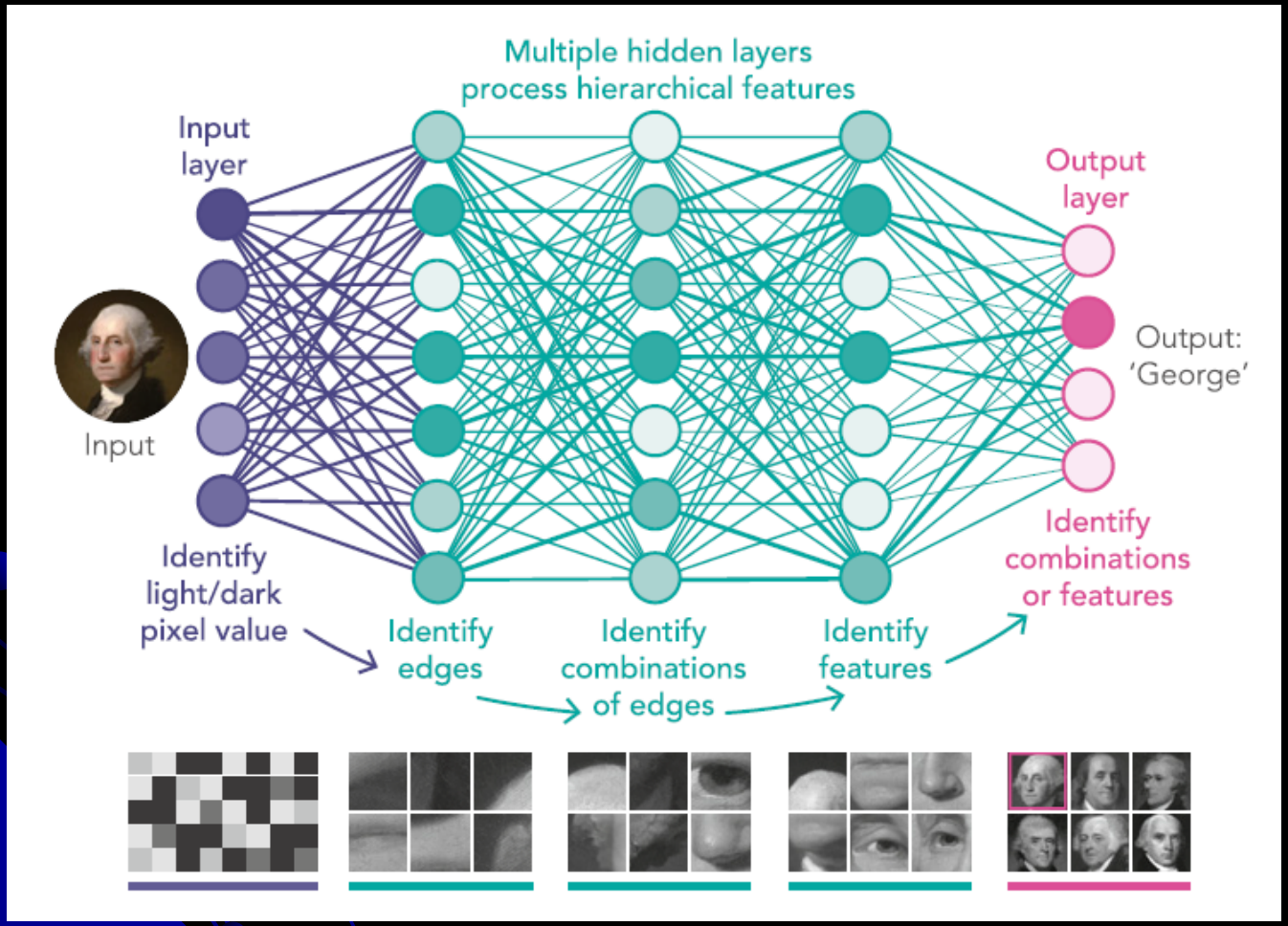
# Number of parameters




[3 x 4] + [4 x 4] + [4 x 1] = 12 + 16 + 4 = 32 weights

4 + 4 + 1 = 9 biases

# Raw image input



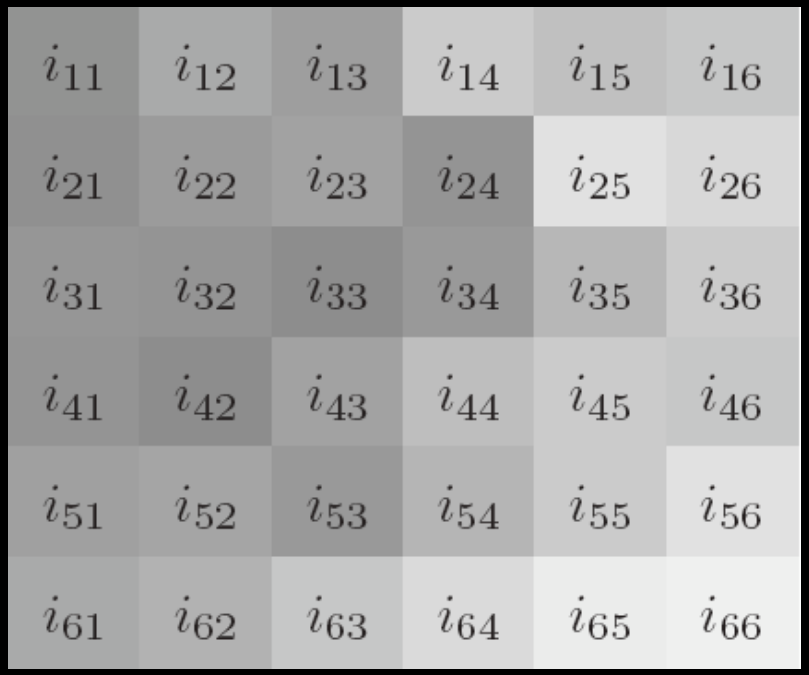

M. Mitchell Waldrop: What are the limits of deep learning?
https://doi.org/10.1073/pnas.1821594116

# Convolutional neural networks

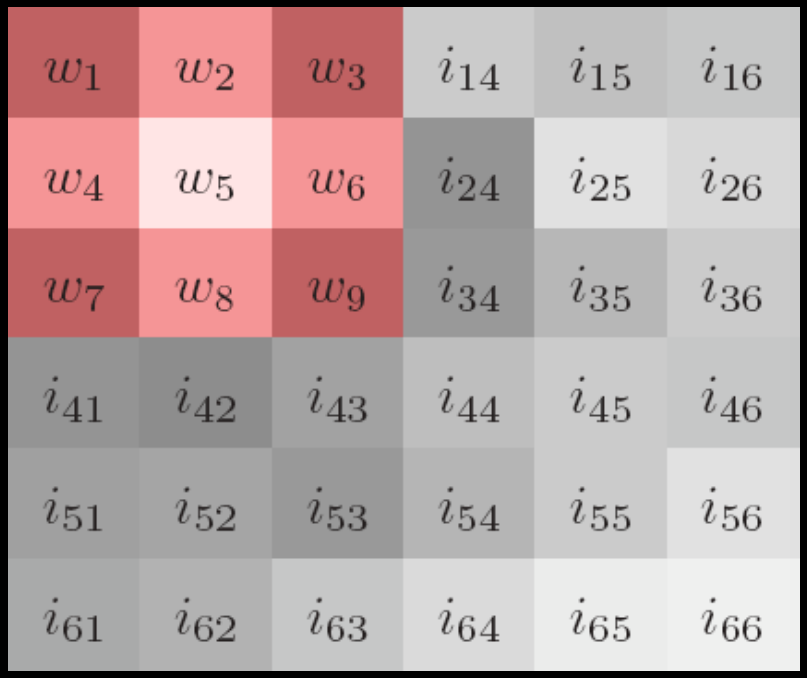
Discrete 2D convolution

$$h(k,l) = (I * w)(k,l)$$
$$= \sum_{m=-M}^{M} \sum_{n=-N}^{N} I(m,n).w(k-m, l-n)$$

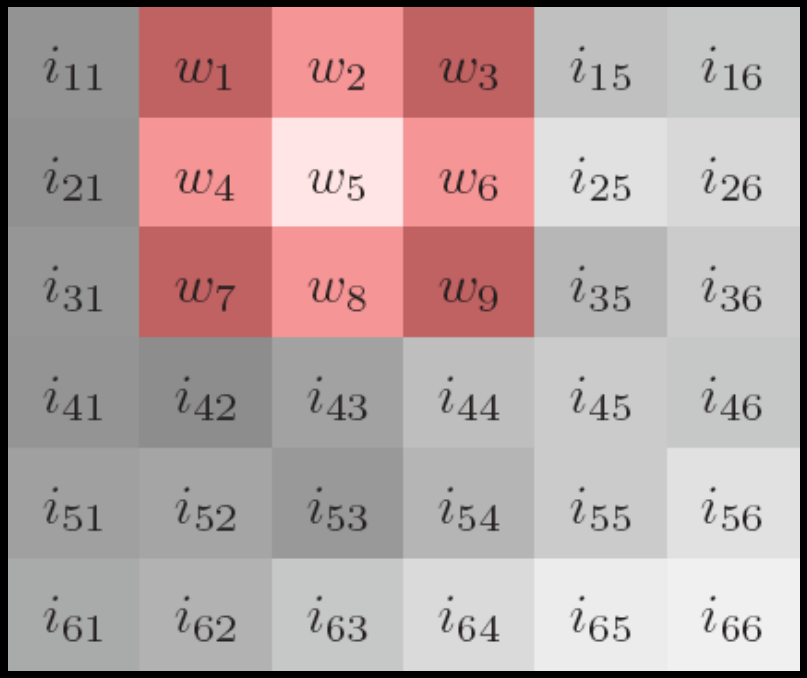
# Convolution 2D



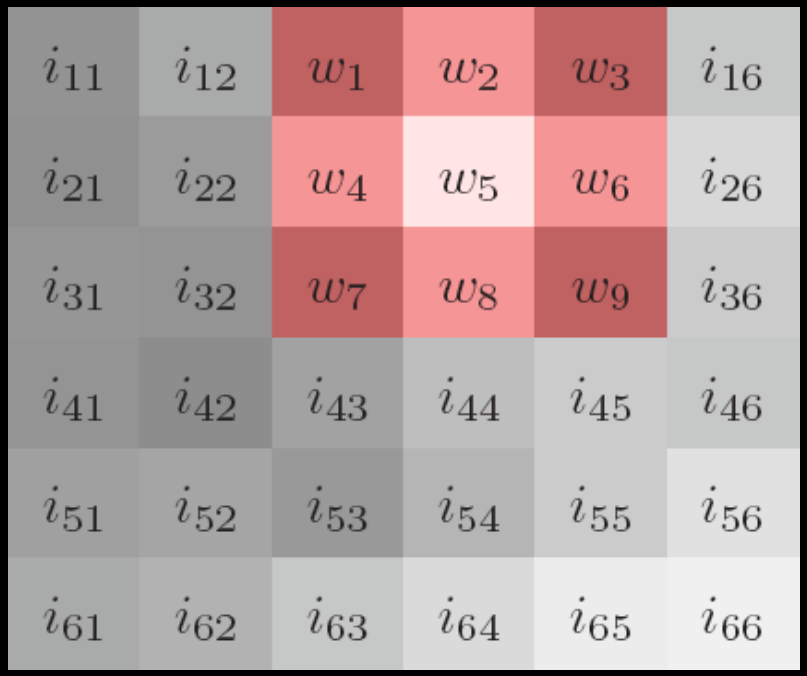
# Convolution 2D



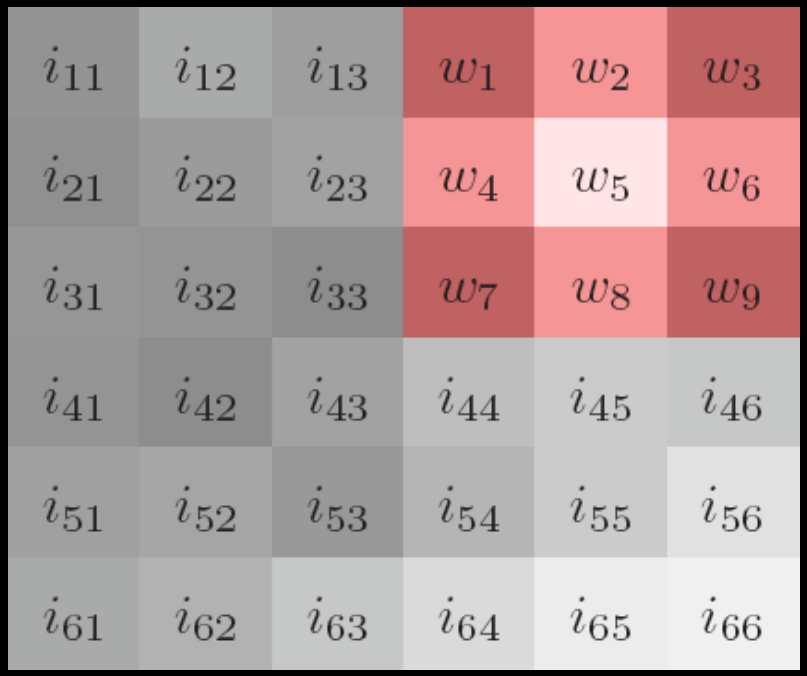
# Convolution 2D



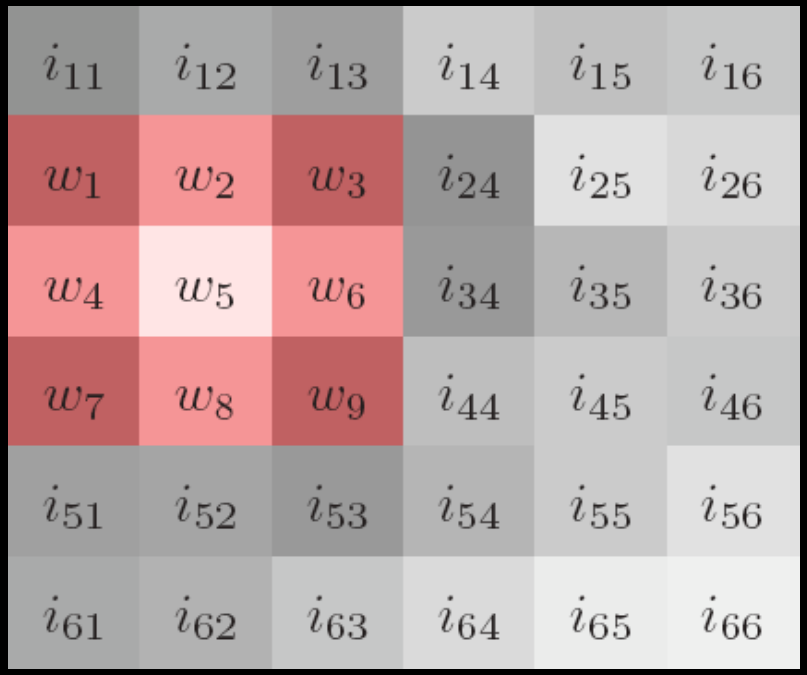
# Convolution 2D



# Convolution 2D



# Convolution 2D

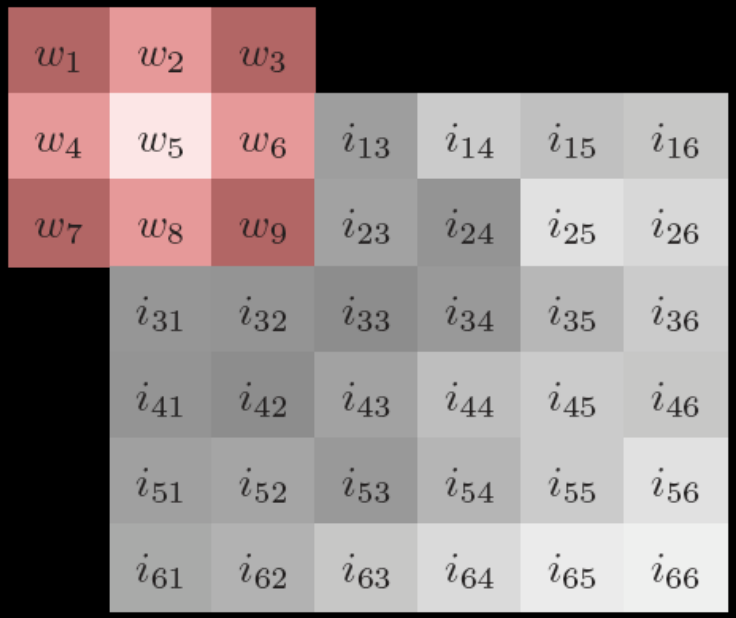


# Convolution 2D

h(0,0) = ?

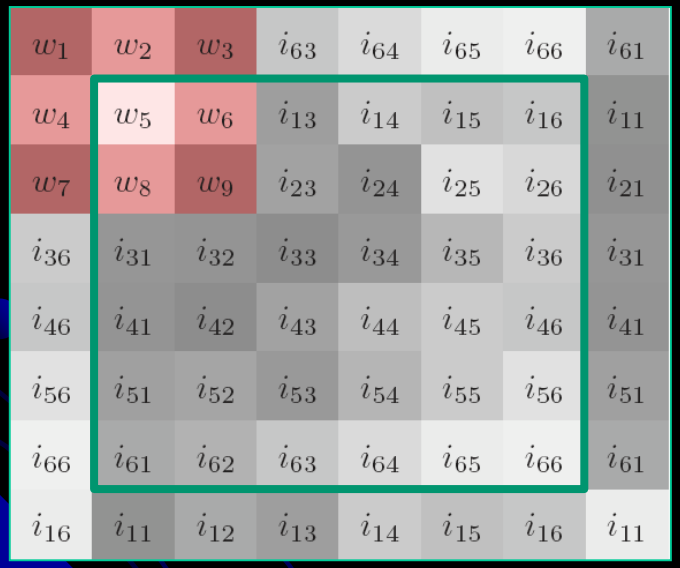
$$h(k,l) = (I * w)(k,l)$$

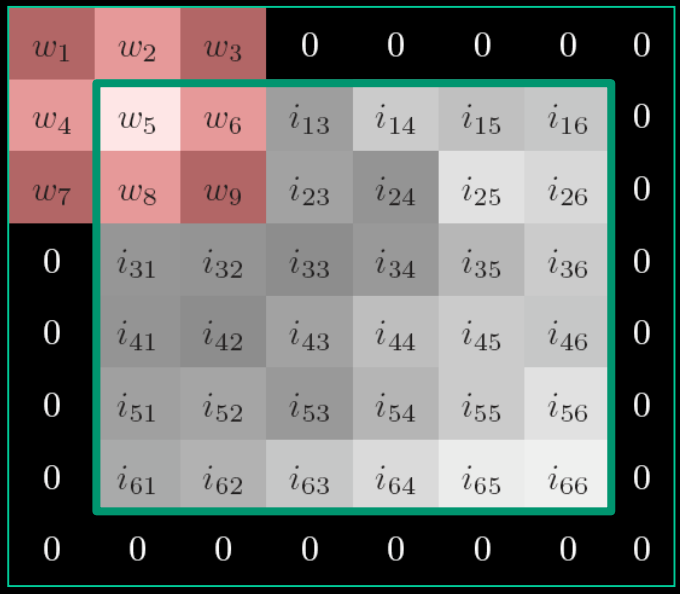
$$= \sum_{m=-M}^{M} \sum_{n=-N}^{N} I(m,n).w(k-m, l-n)$$



# Convolution 2D

Border padding



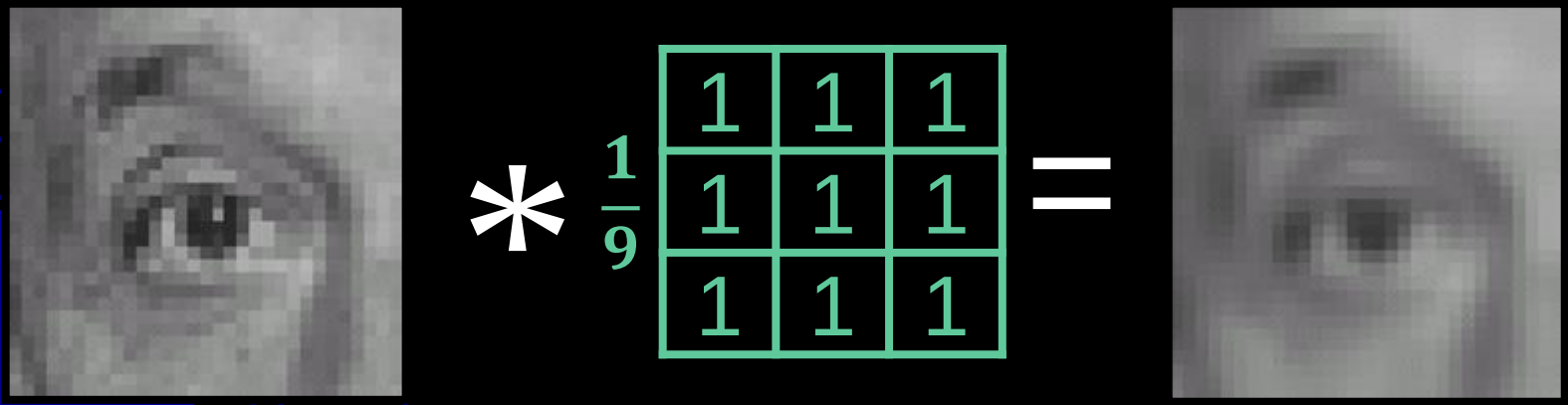



# Kernels used in CV

Edge detection

Blurring

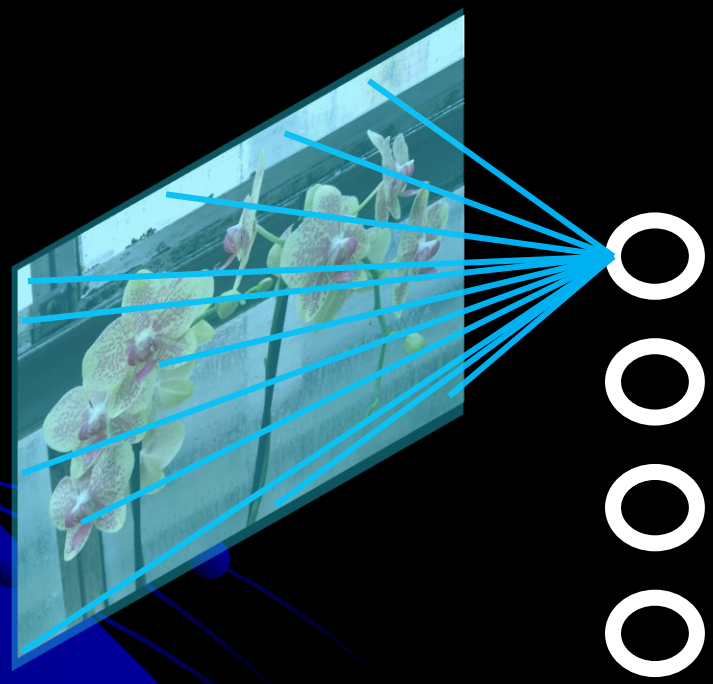
Sharpening

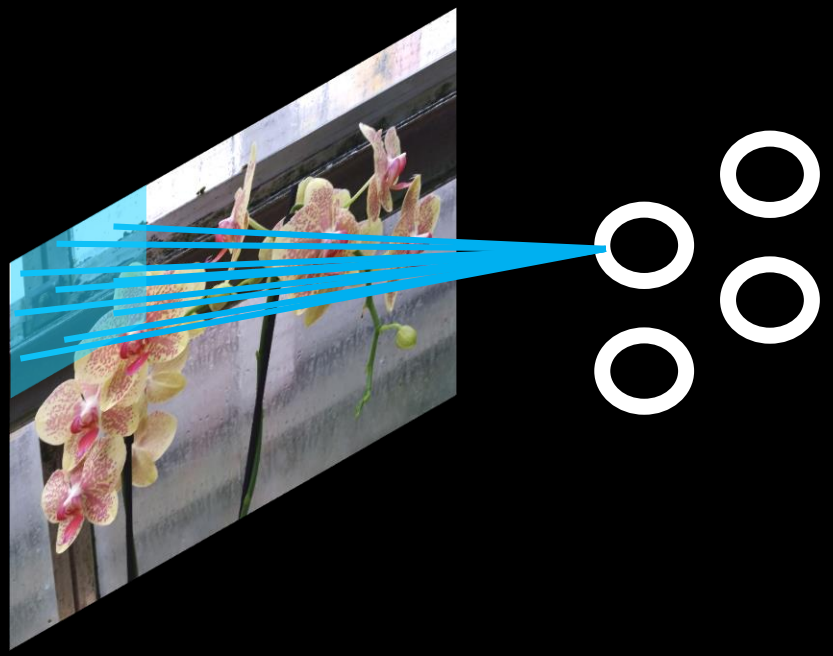



Original

Blur (with a mean filter)

# Convolutional neural networks

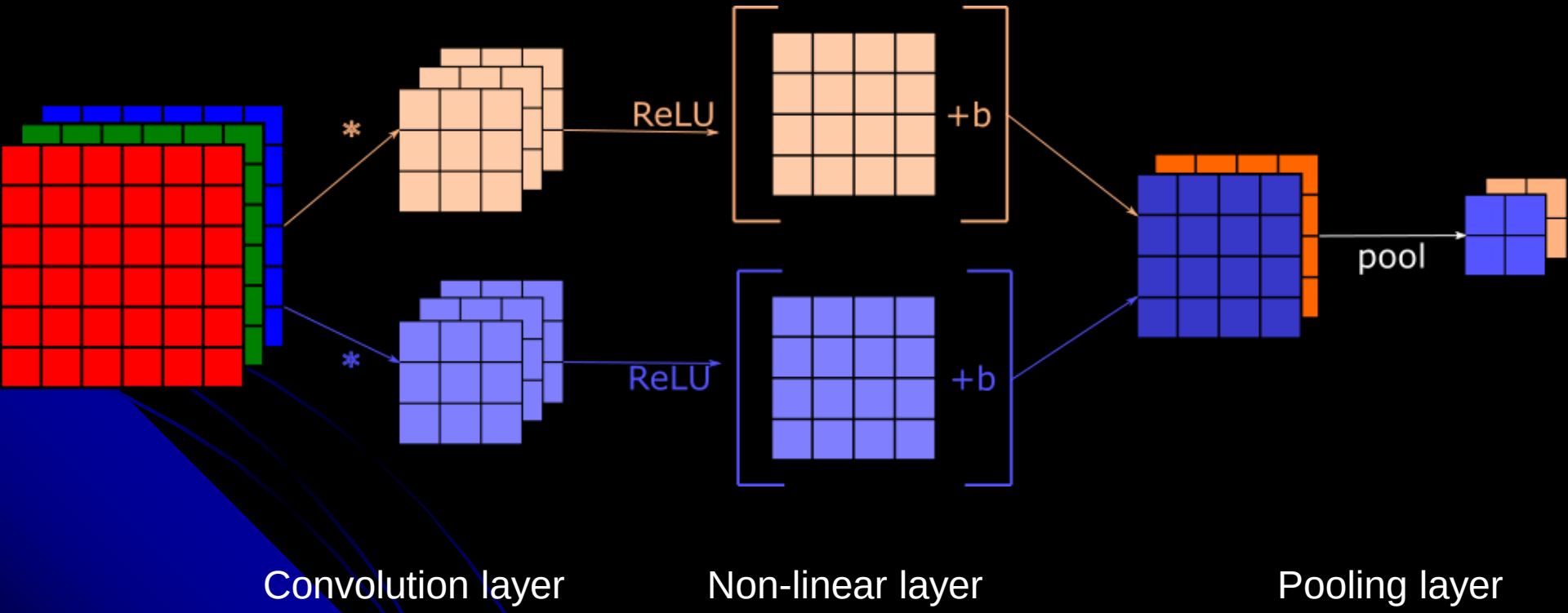




$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

$$h(k,l) = (I * w)(k,l)$$

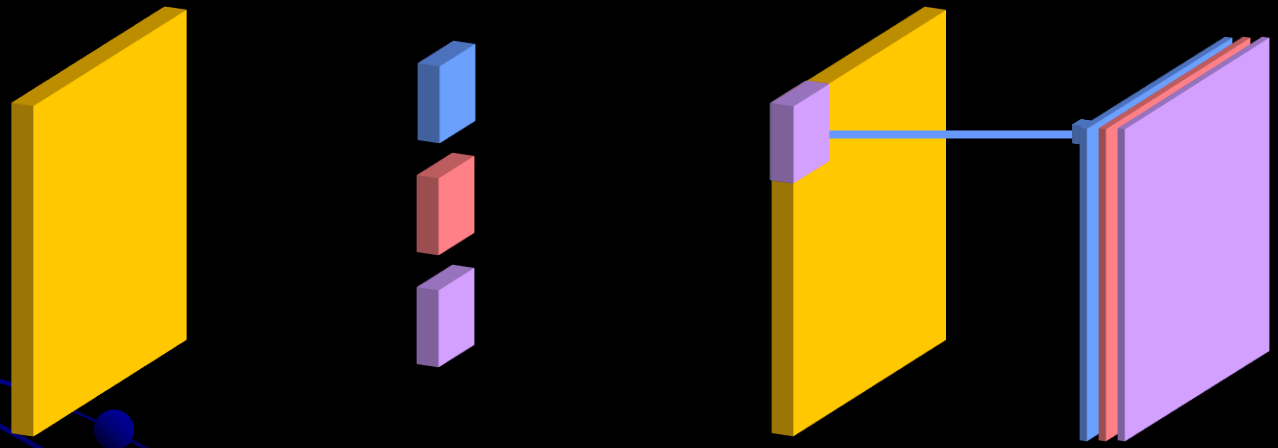
# Convolutional neural networks




Convolution layer Non-linear layer Pooling layer

# Convolution layer

Image 32x32x3  Filters 5x5x3

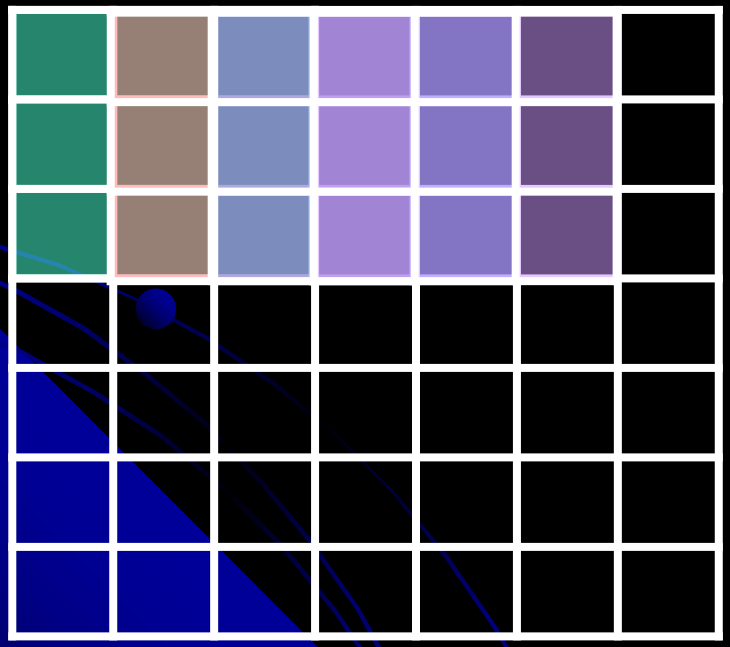


1x1xD filter size?

# Convolution layer

Stride – next filter position

Size of output: (N-F)/stride+1



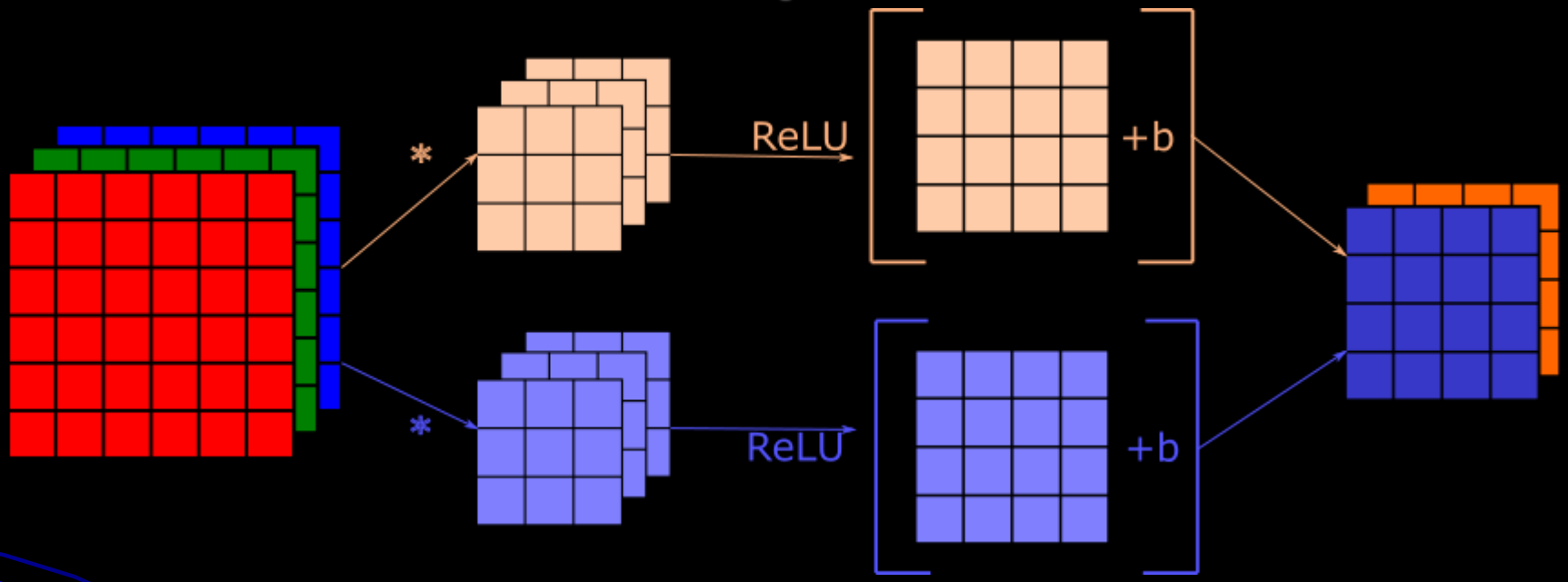
stride=1 → 5

stride=2 → 3

stride=3 → 2.33

(N-F+2P)/stride+1

# Number of parameters




Weights and biases:

2 x [3 x 3 x 3] = 54 weights

2 biases

http://cs231n.github.io/neural-networks-1/

# Non-linear layer

Activation function

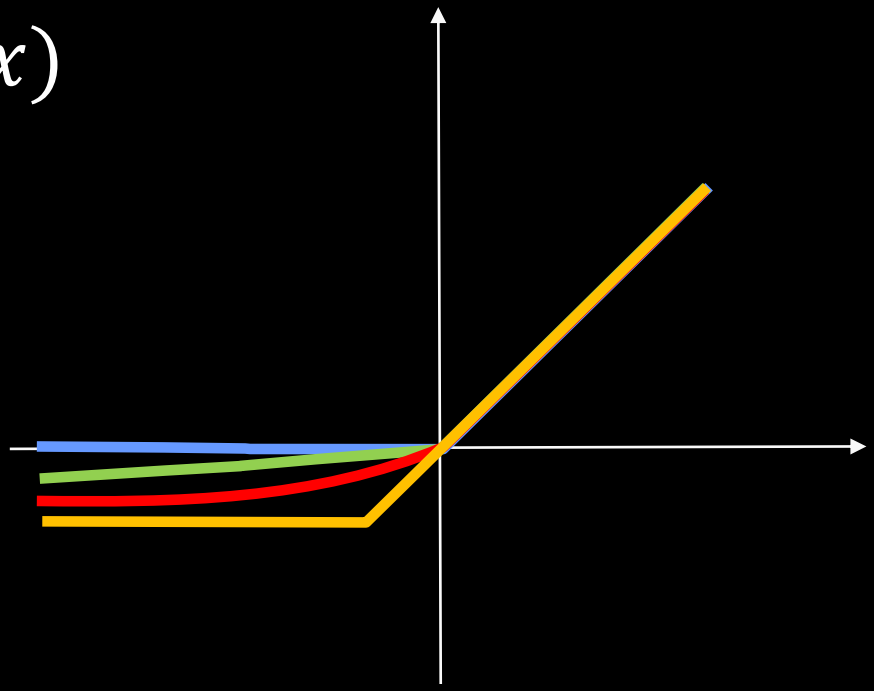
ReLU $f(x) = \max(0, x)$

And modifications:

Leaky ReLU,

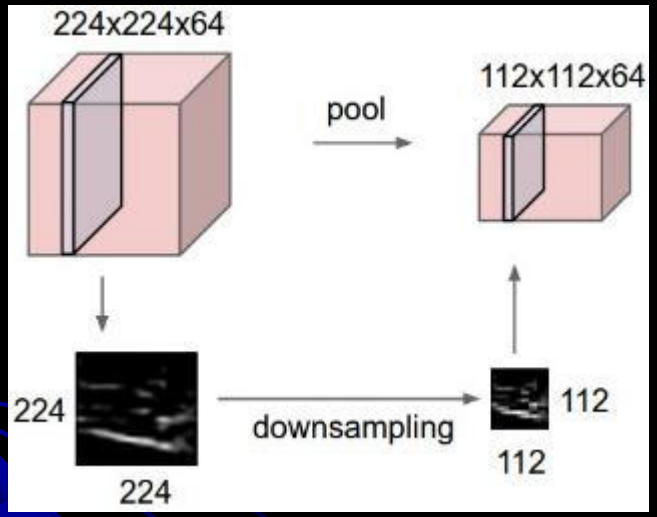
ELU,

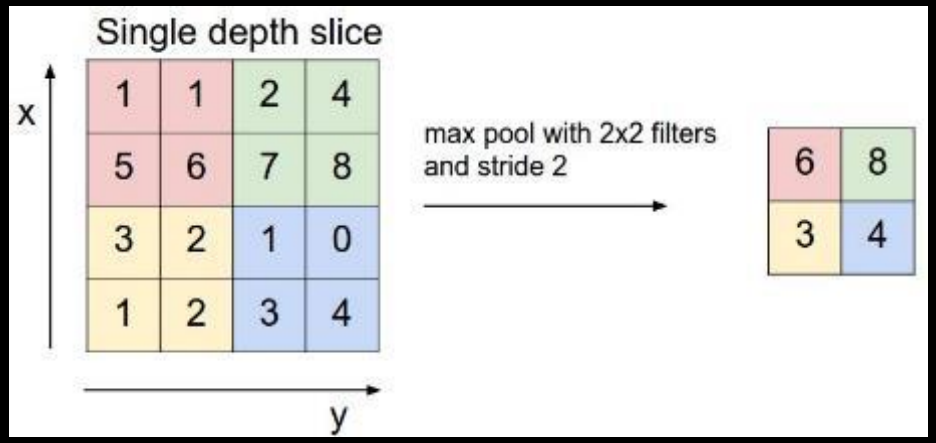
SReLU…



# Pooling layer

Decreases size, nr. of parameters



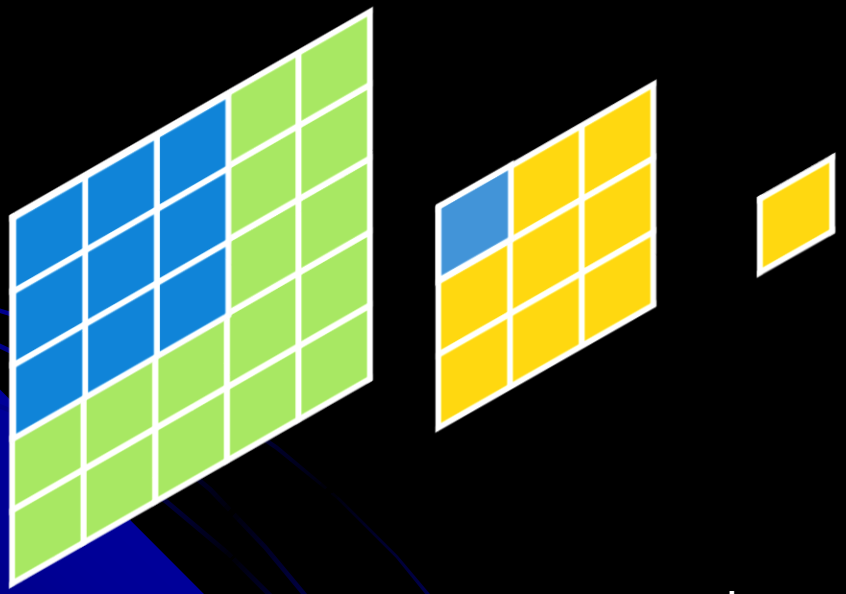




Max, average, sum

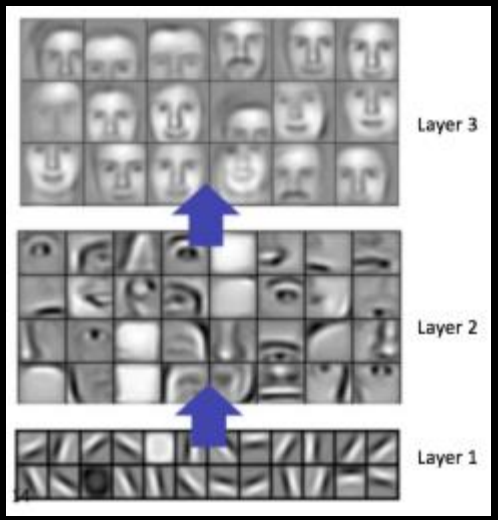
Size, stride

# Receptive field

Input area of a neuron

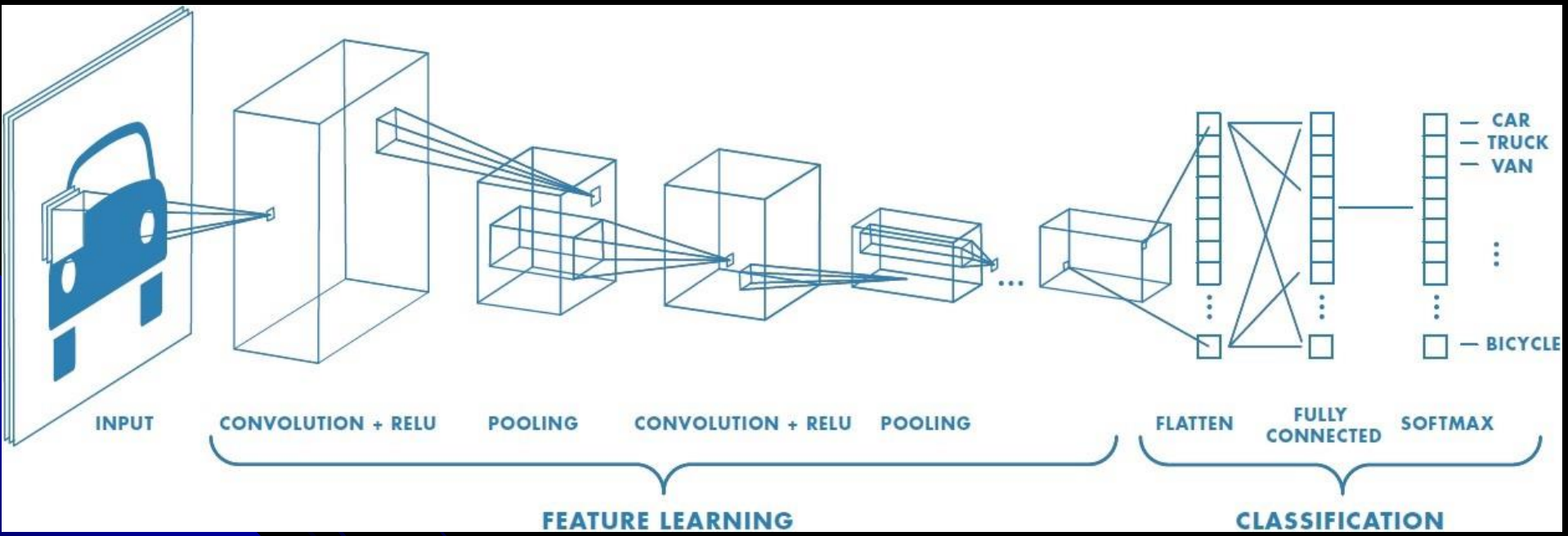


Layer 1 Layer 2 Layer 3




https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

# Sample architecture




https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

# Alexnet

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

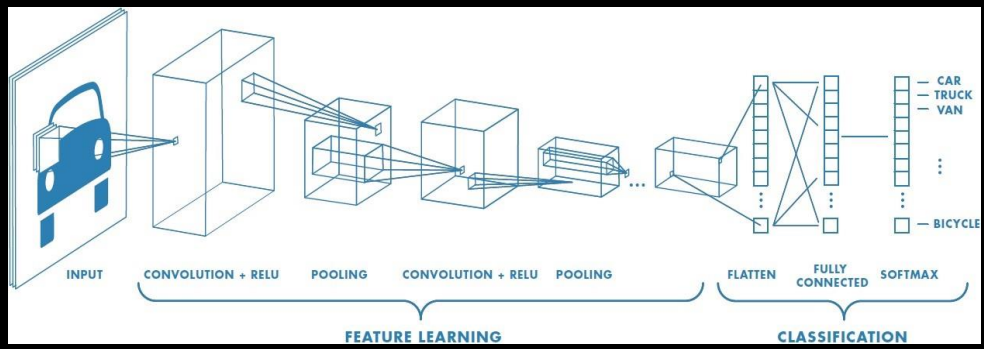
[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

# CNN usage I

Classifier:
classify new images

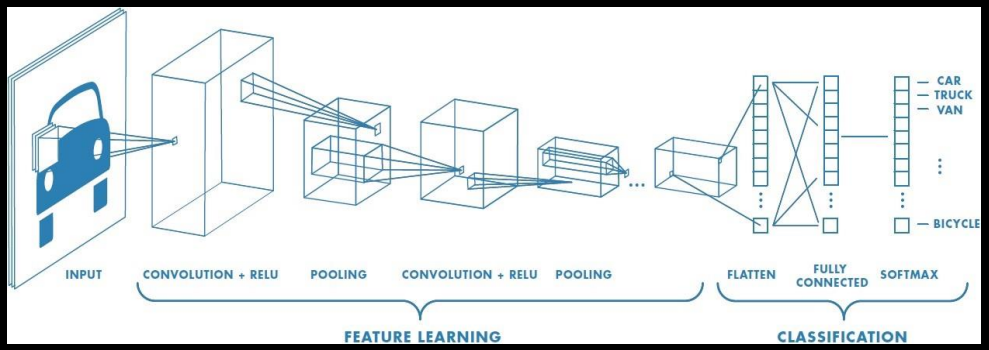



Standalone Feature Extractor:
pre-process images and extract relevant features.

https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/

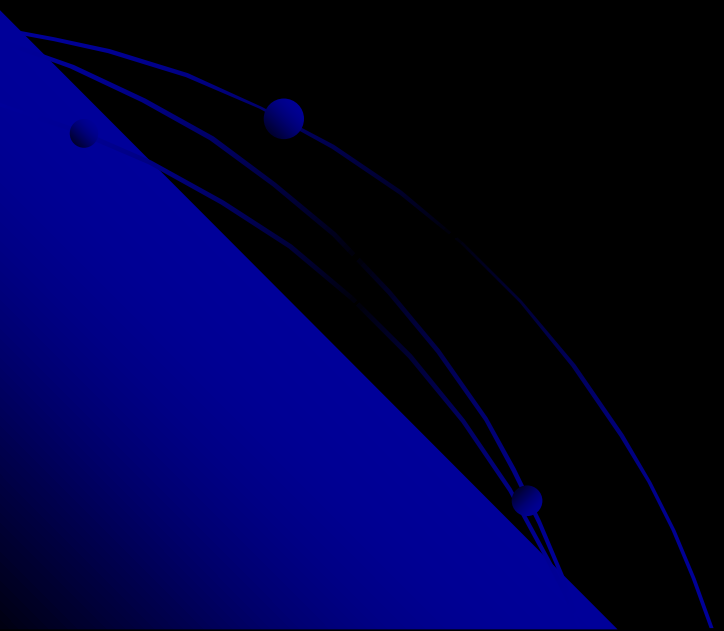
# CNN usage II

Integrated Feature Extractor:
integrated into a new model, but layers of the pre-trained model are frozen during training




- Weight Initialization:
integrated into a new model, and the layers of the pre-trained model are trained with the new model

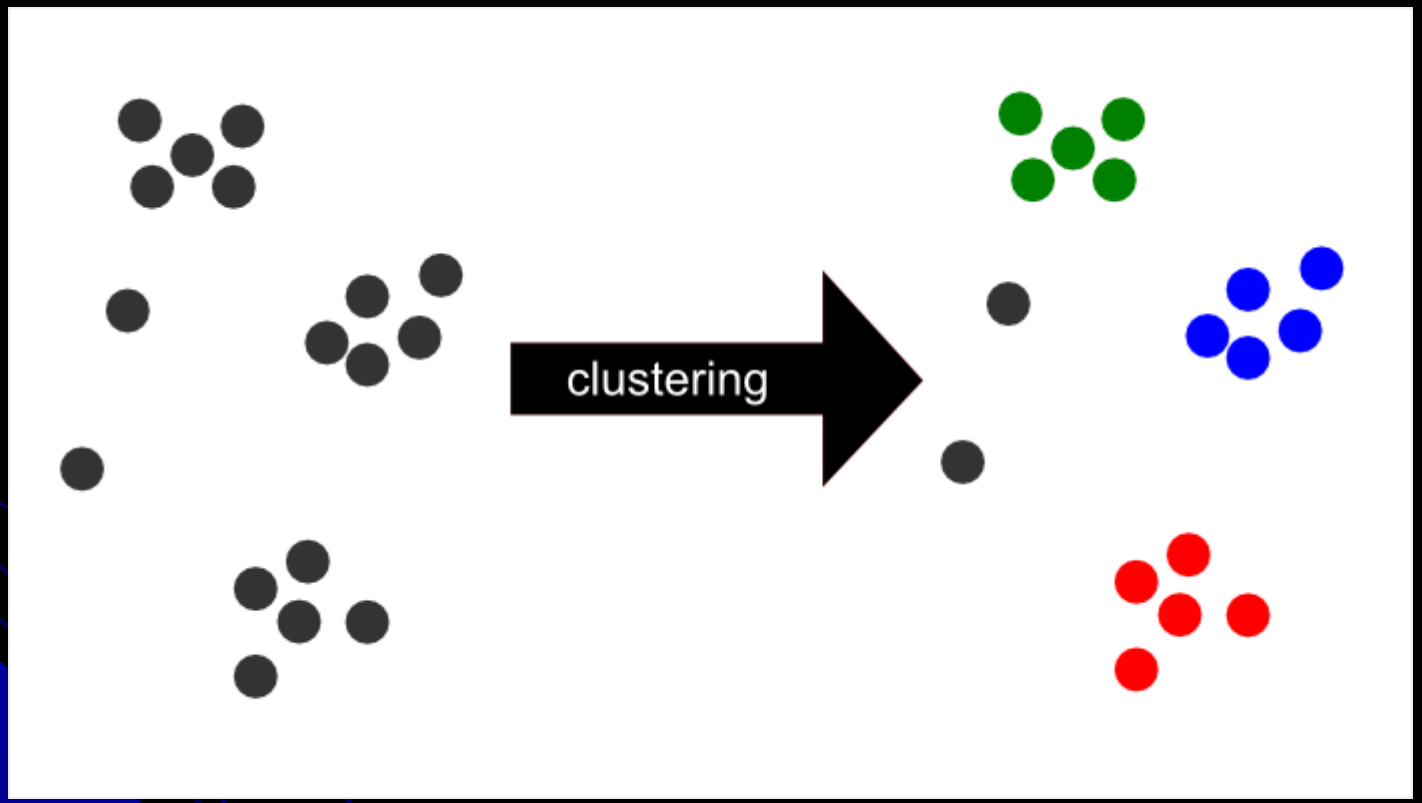
# Unsupervised methods



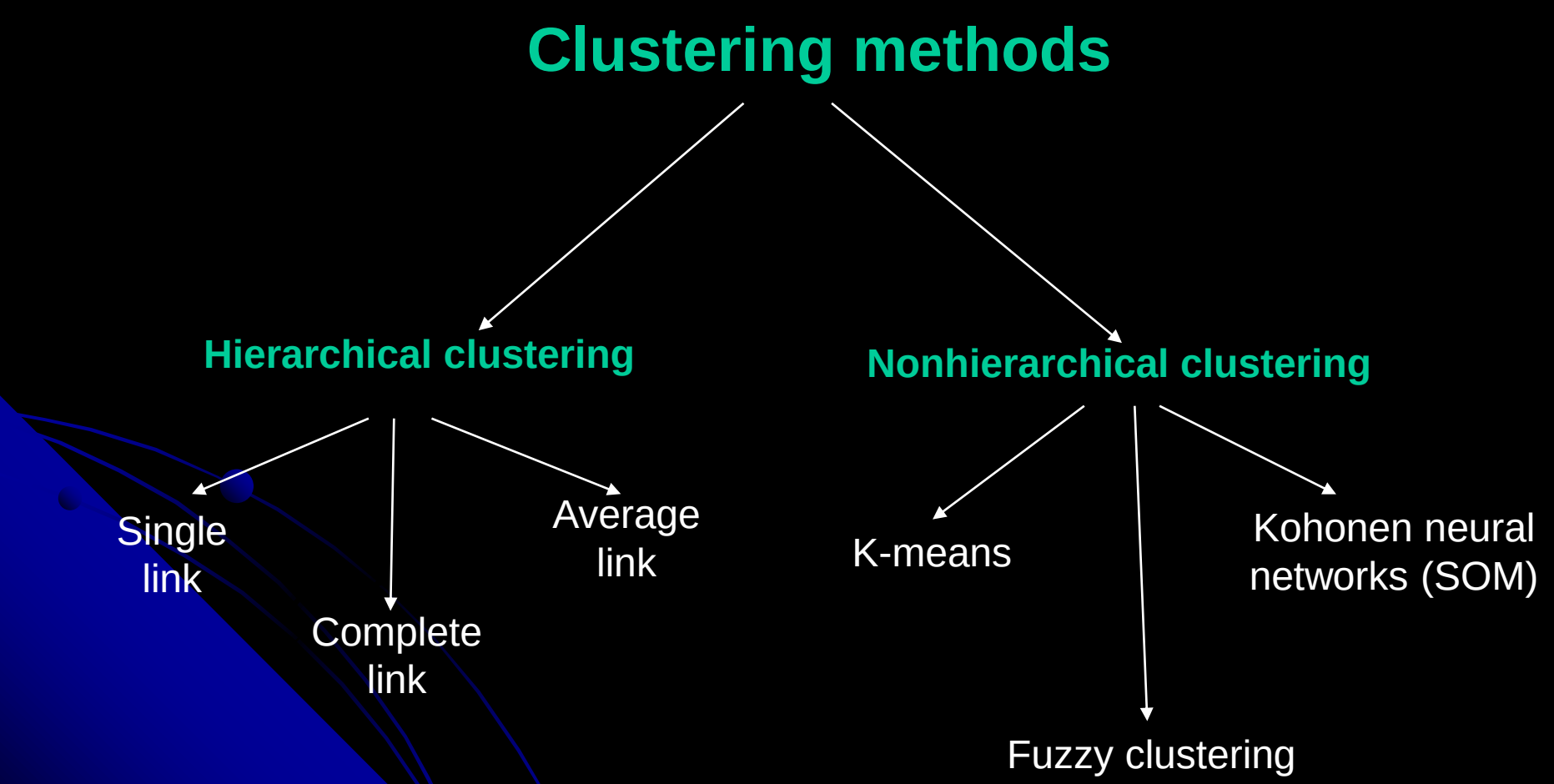
# Unsupervised methods

- Definition 1
  - Supervised: human effort involved
  - Unsupervised: no human effort
- Definition 2
  - Supervised: learning conditional distribution P(Y|X), X: features, Y: classes
  - Unsupervised: learning distribution P(X), X: features

# Unsupervised methods

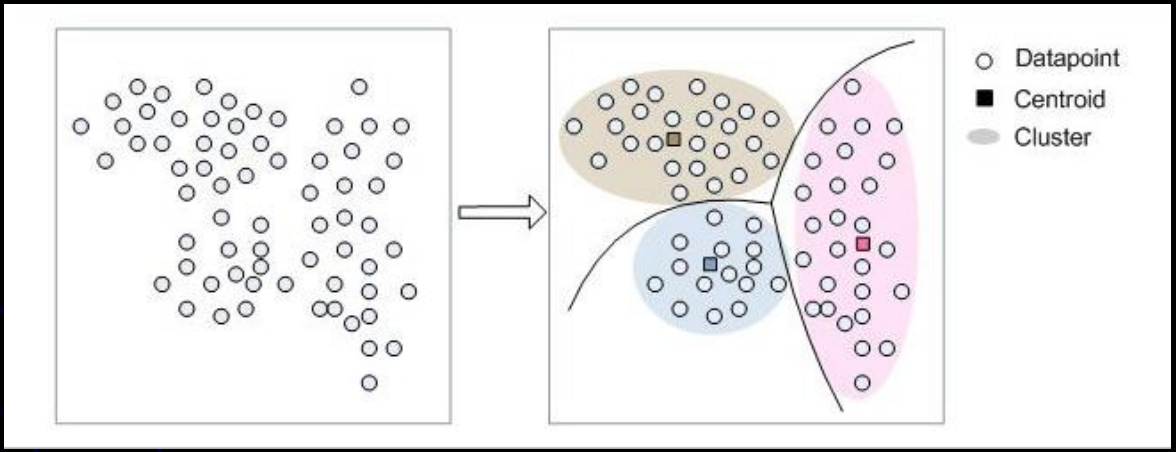


# Unsupervised methods



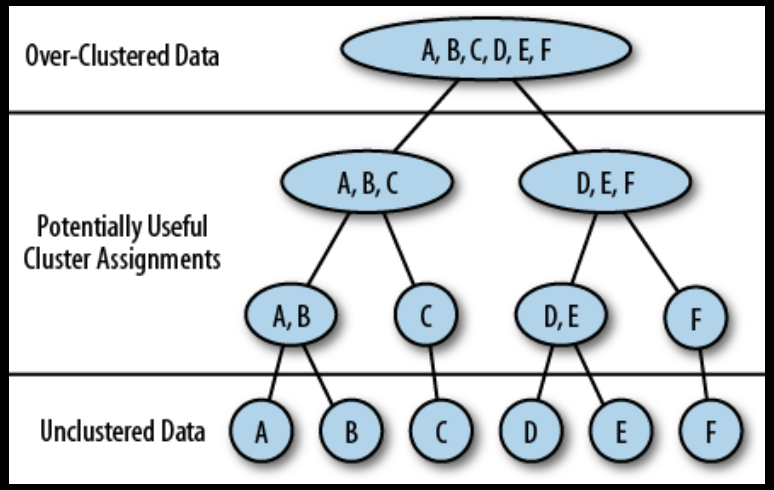
# Clustering

Nonhierarchical – the space is divided into one set of clusters



# Clustering

Hierarchical – levels of clusters form a tree structure



# Clustering

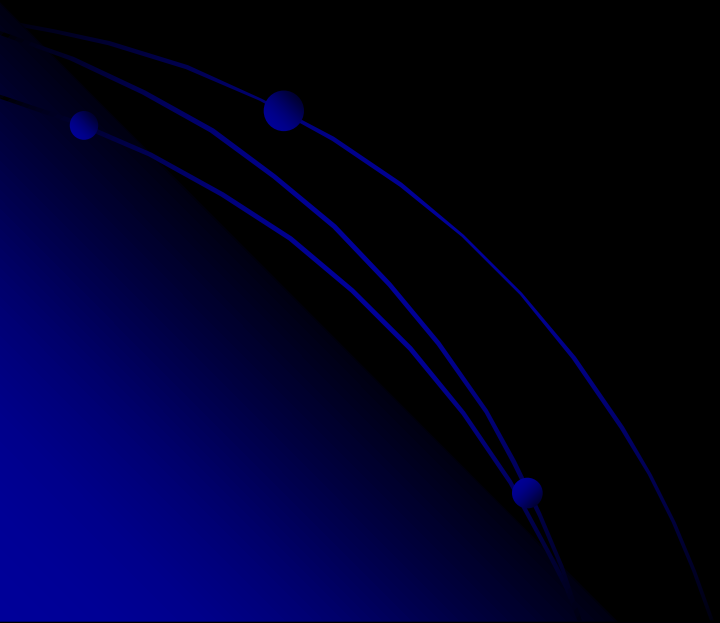
2 approaches

Agglomerative (Bottom-up)
Divisive (Top-down)

# Clustering

- Hard vs. Soft
  - Hard: same object can only belong to single cluster
  - Soft: same object can belong to different clusters
    - E.g. Gaussian mixture model

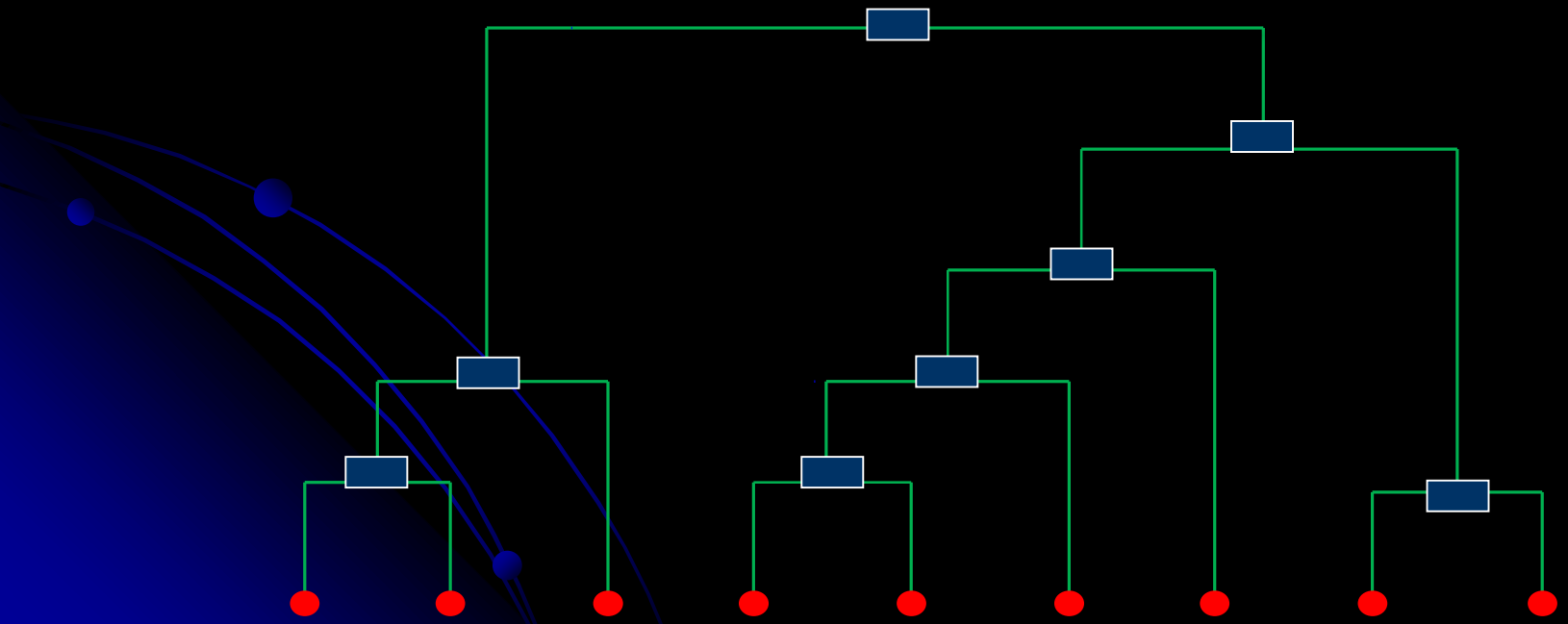
# Hierarchical methods



# Dendrogram

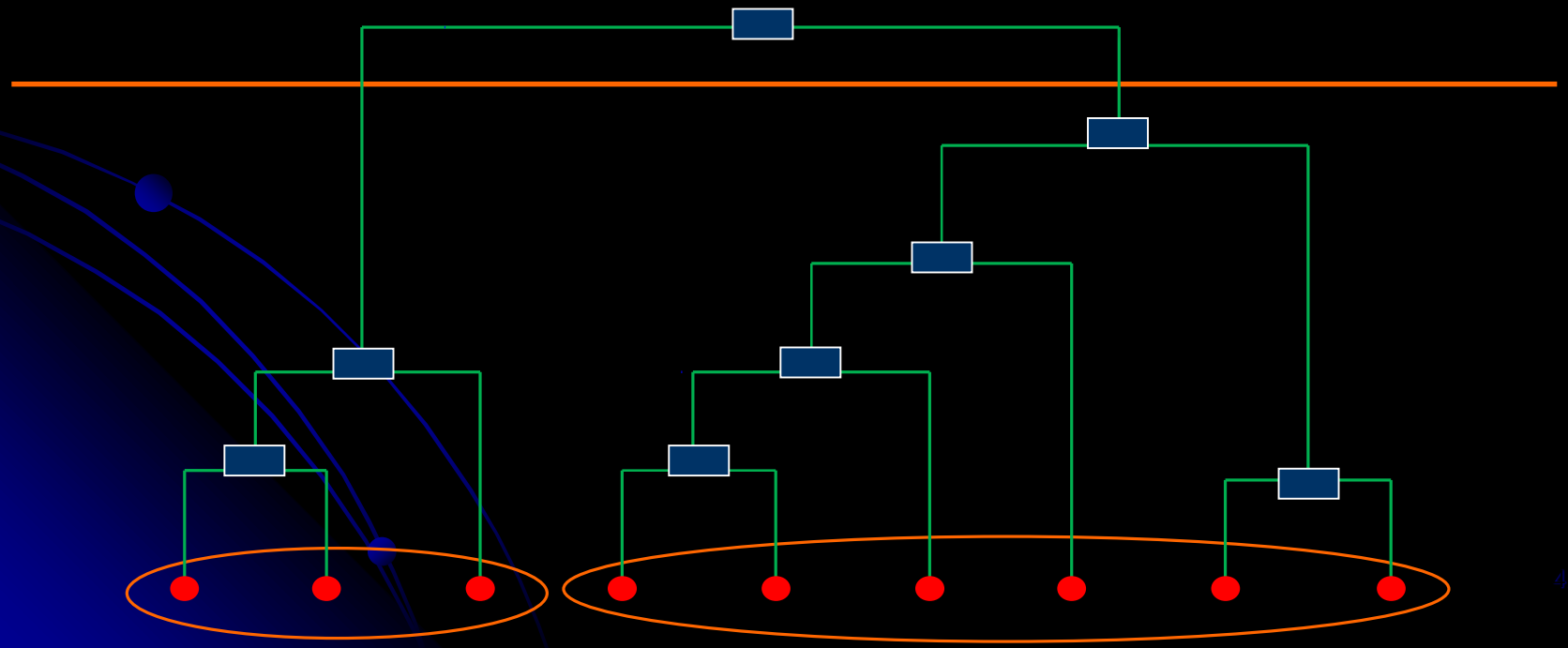
A binary tree that shows how clusters are merged/split hierarchically

Each node on the tree is a cluster; each leaf node is a singleton cluster



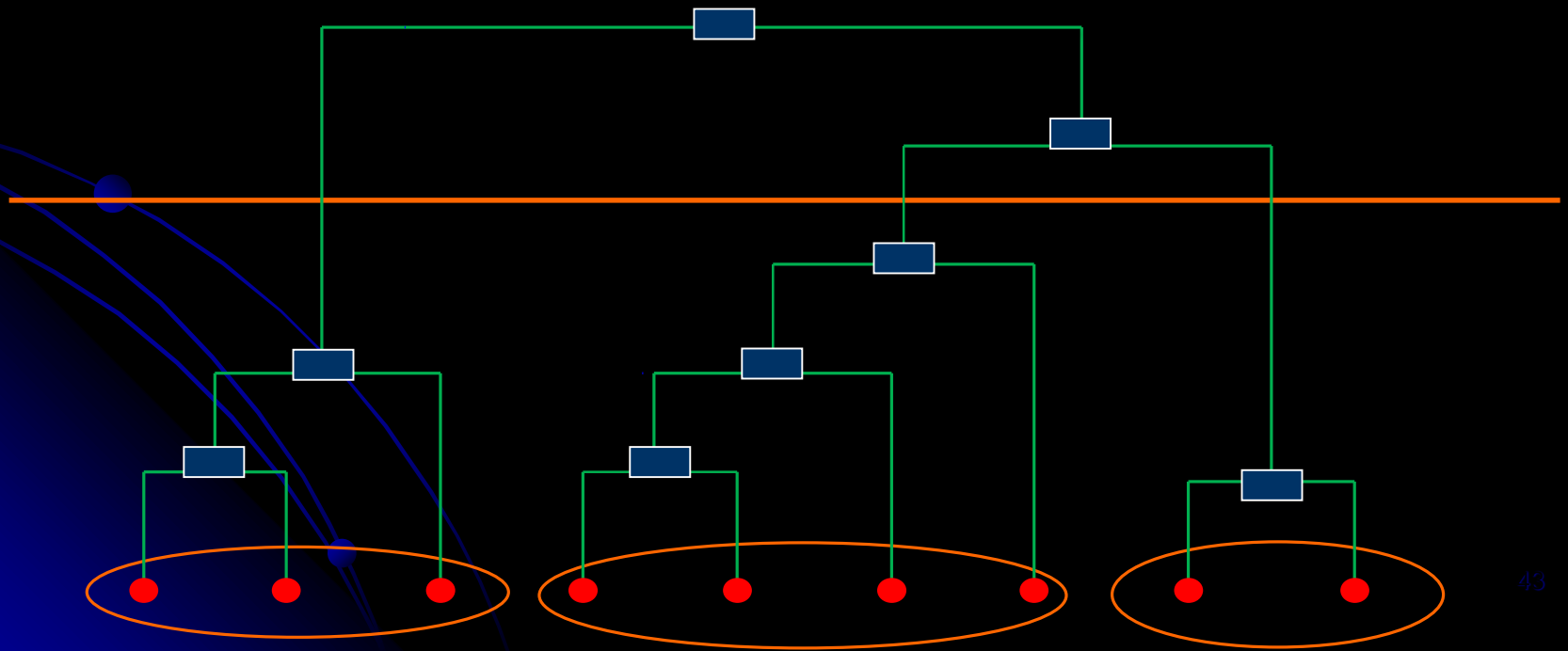
# Dendrogram

A clustering of the data objects is obtained by cutting the *dendrogram* at the desired level, then each connected component forms a cluster



# Dendrogram

A clustering of the data objects is obtained by cutting the *dendrogram* at the desired level, then each connected component forms a cluster



# Divisive approach

Put all object into one cluster
REPEAT
    Pick the cluster to split
    Split the cluster
UNTIL individual objects

or desired number of clusters

# DIANA (Divisive Analysis)

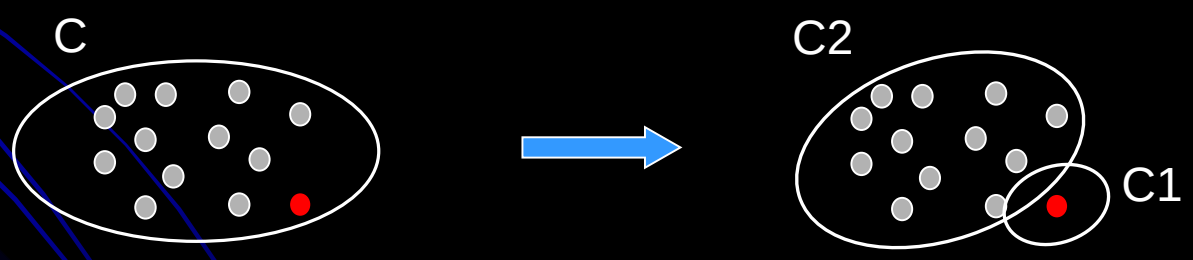
Introduced in Kaufmann and Rousseeuw (1990)

Basic algorithm for divisive clustering

# Splitting Process of DIANA

Initialization:

1. Choose the object O which is most dissimilar to other objects in C
2. Let C1={O}, C2=C-C1

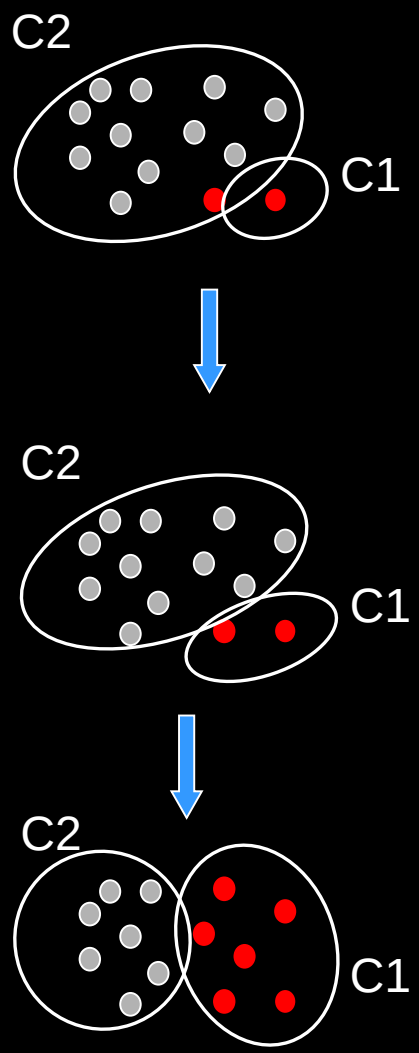


# Splitting Process of DIANA

3. For each object $O_i$ in C2, compare average distances to C1 and C2:

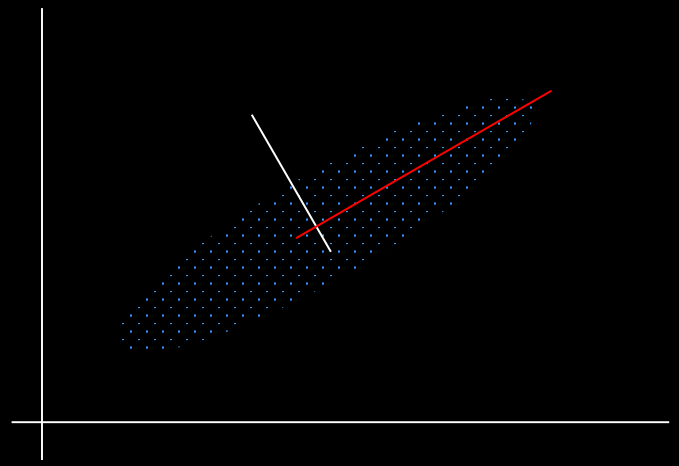
$$D_i = \underset{O_j \in C2}{\text{avg}}\ d(O_i, O_j) - \underset{O_j \in C1}{\text{avg}}\ d(O_i, O_j)$$

4. Choose the object $O_k$ with greatest D score
5. If $D_k > 0$, move $O_k$ from C2 to C1, and repeat 3-5.
6. Otherwise, stop splitting process.

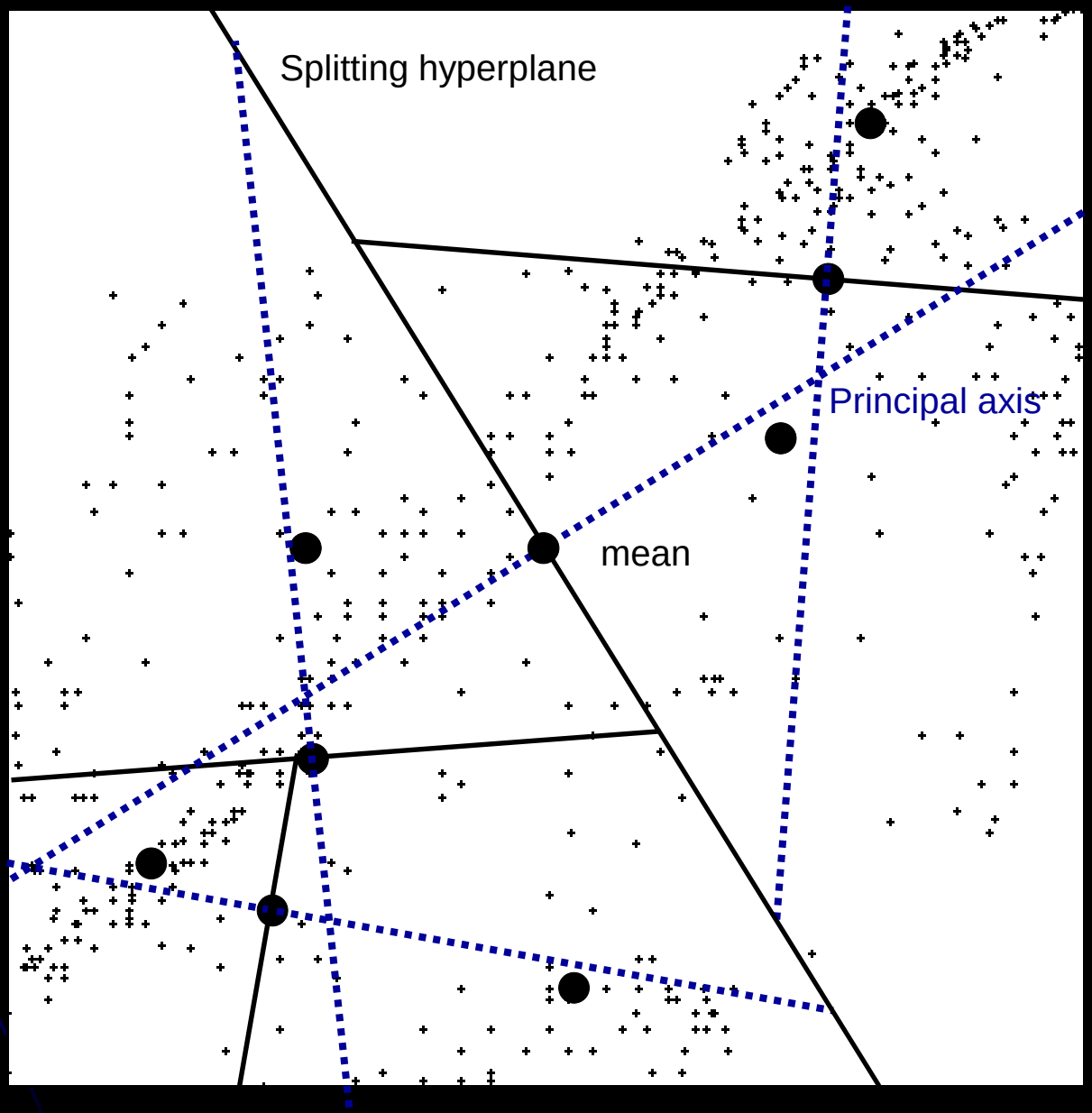


# Principal Directions Divisive Partitioning

1. Find the principal axis
2. Split point at mean projection
3. Find new centroids

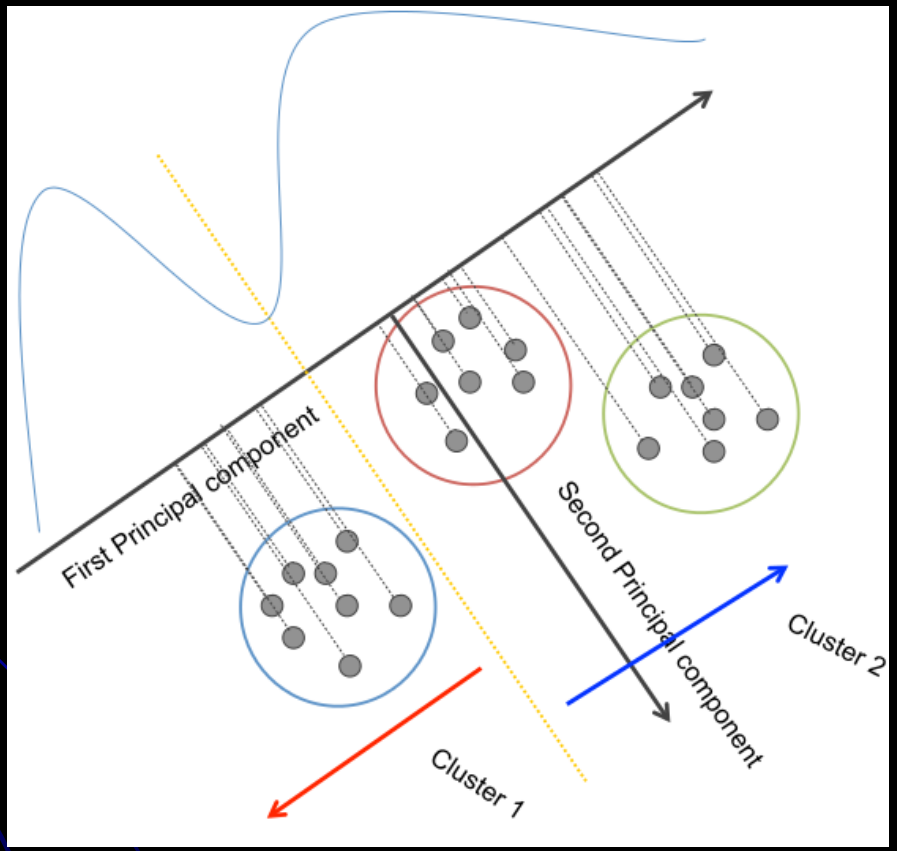


1. Find the principal axis
2. Split point at mean projection
3. Find new centroids



# Principal Directions Divisive Partitioning

Improvements exist




Tasoulis, S. K., Tasoulis, D. K., and Plagianakos, V. P. (2010). Enhancing principal direction divisive clustering. Pattern Recognition, Vol. 43,

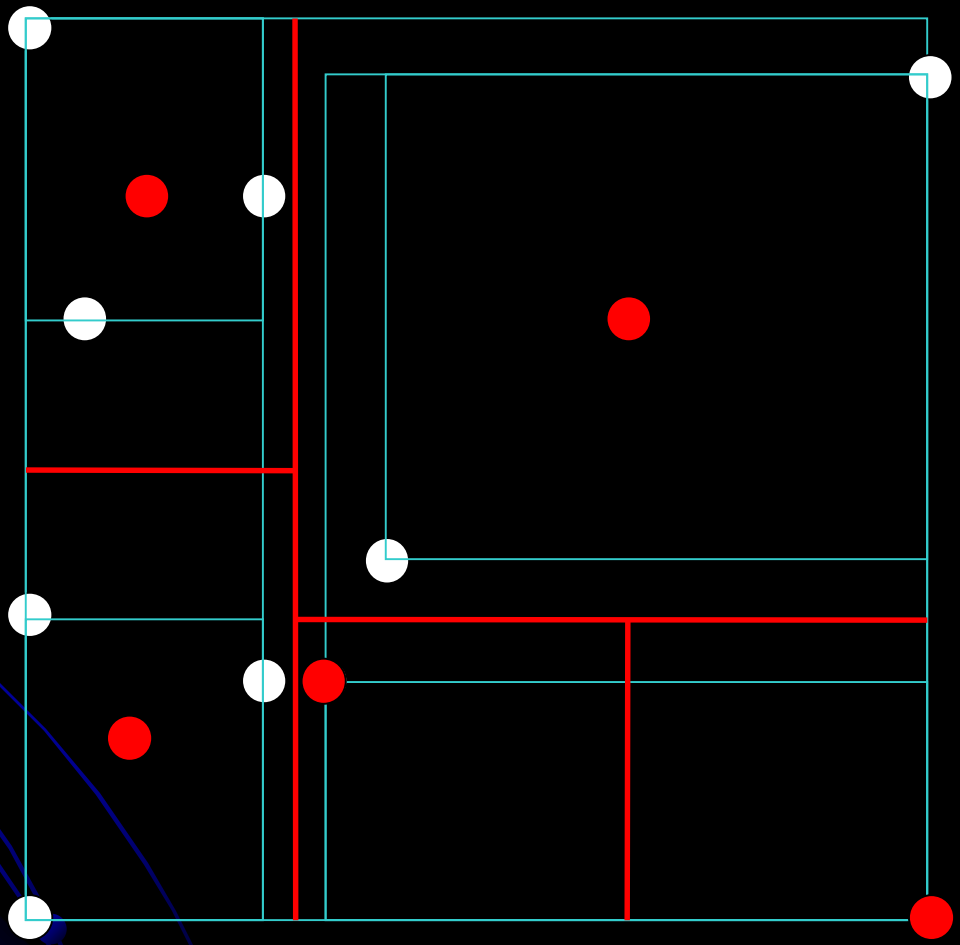
# Median cut

Idea – the prototypes of clusters represent the same number of objects

The prototype = the mean

1. Find the smallest box which contains all the objects
2. Sort the enclosed objects along the longest axis of the box (variance)
3. Split the box into 2 regions at median of the sorted list
4. Repeat the above process until the space has been divided into K regions

# Median Cut



4

32

8

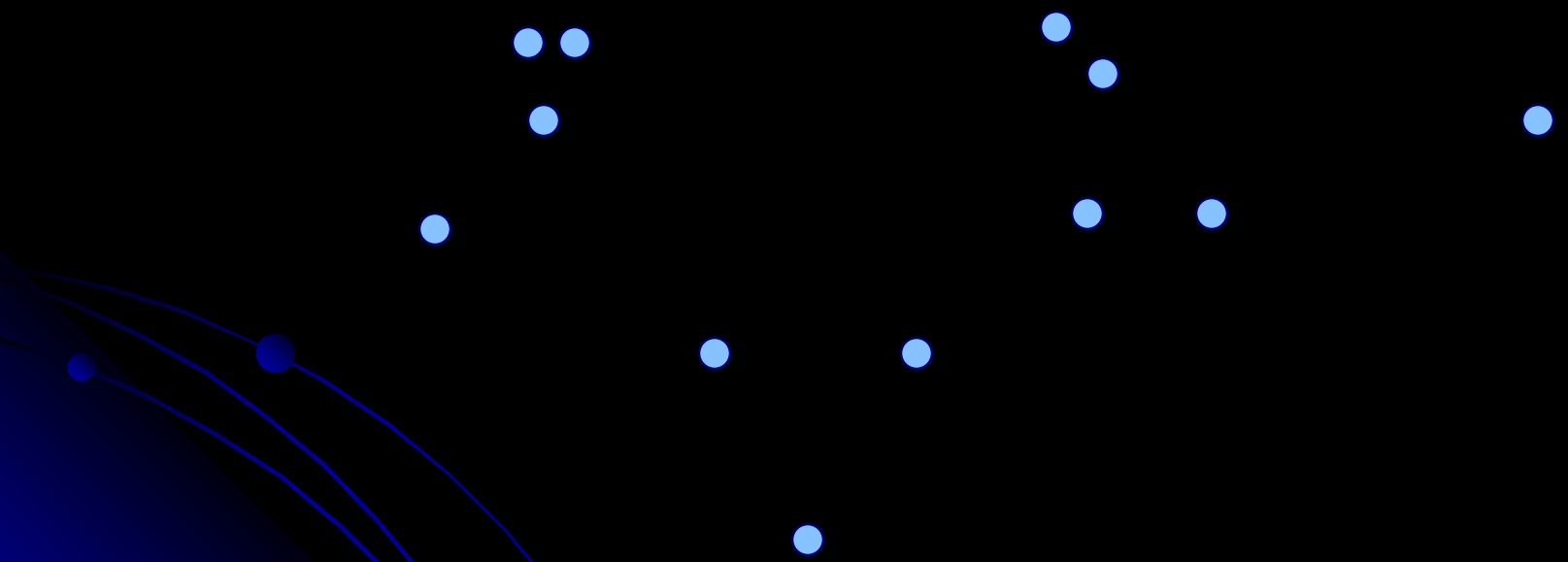
256

# Agglomerative clustering

Individual objects form clusters
REPEAT
    Merge 2 closest clusters (depends on metrics)
UNTIL 1 cluster containing all objects

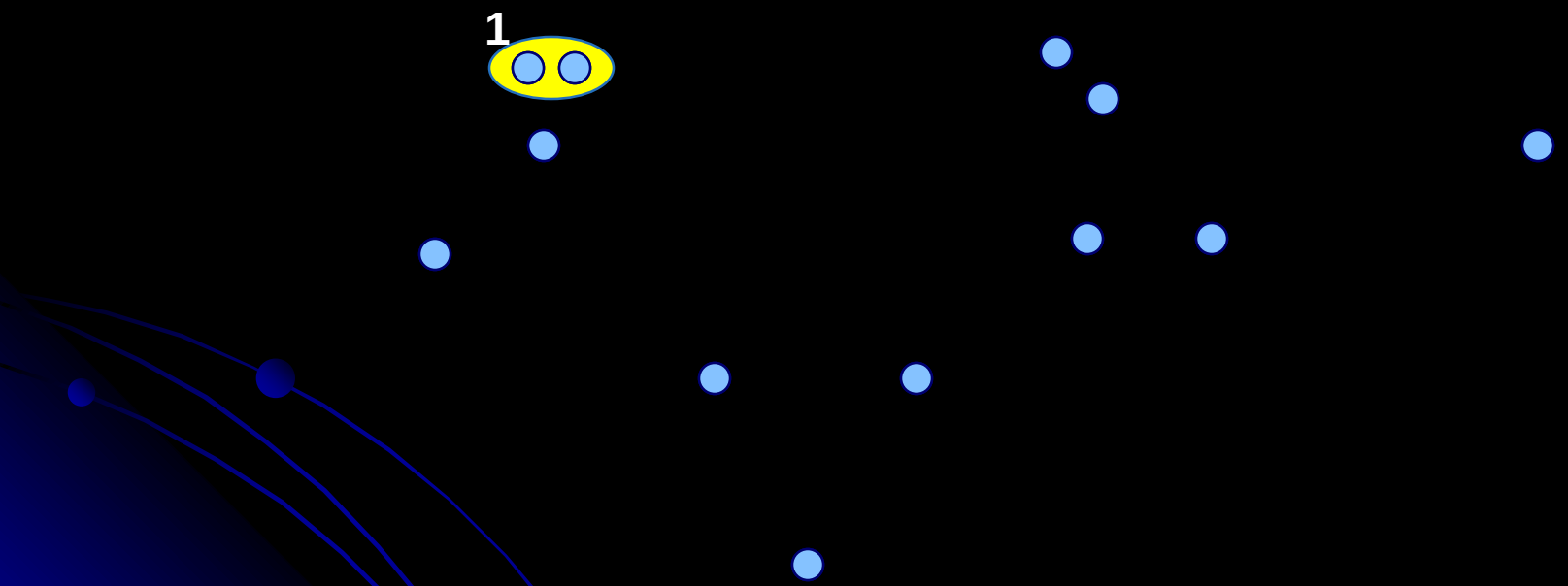
or desired number of clusters

# Agglomerative clustering



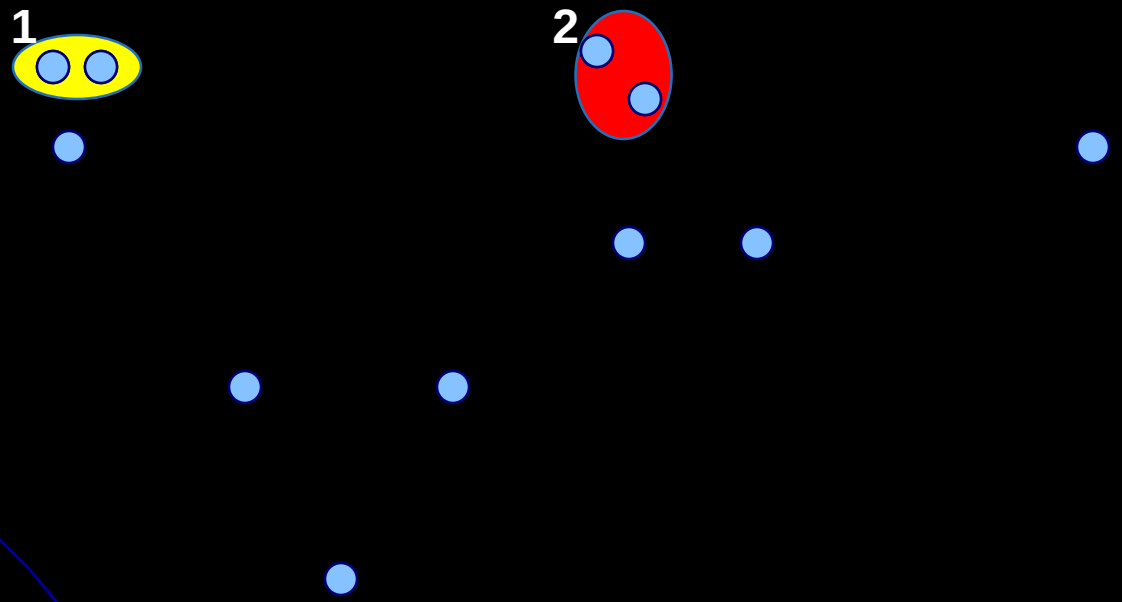
# Agglomerative clustering

1st iteration



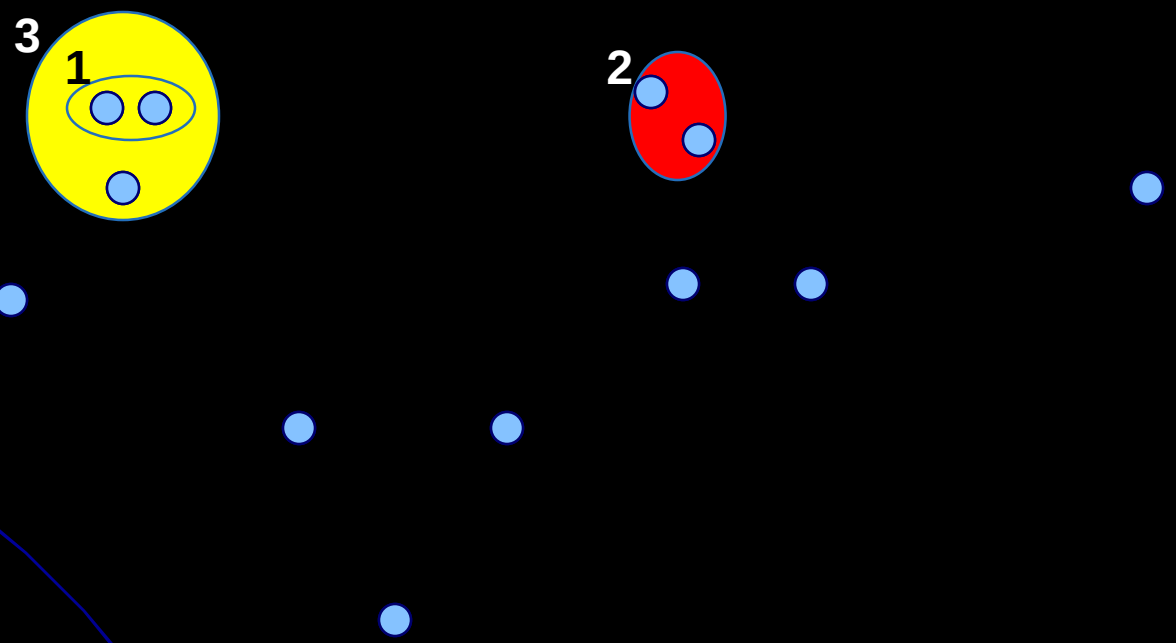
# Agglomerative clustering

2nd iteration



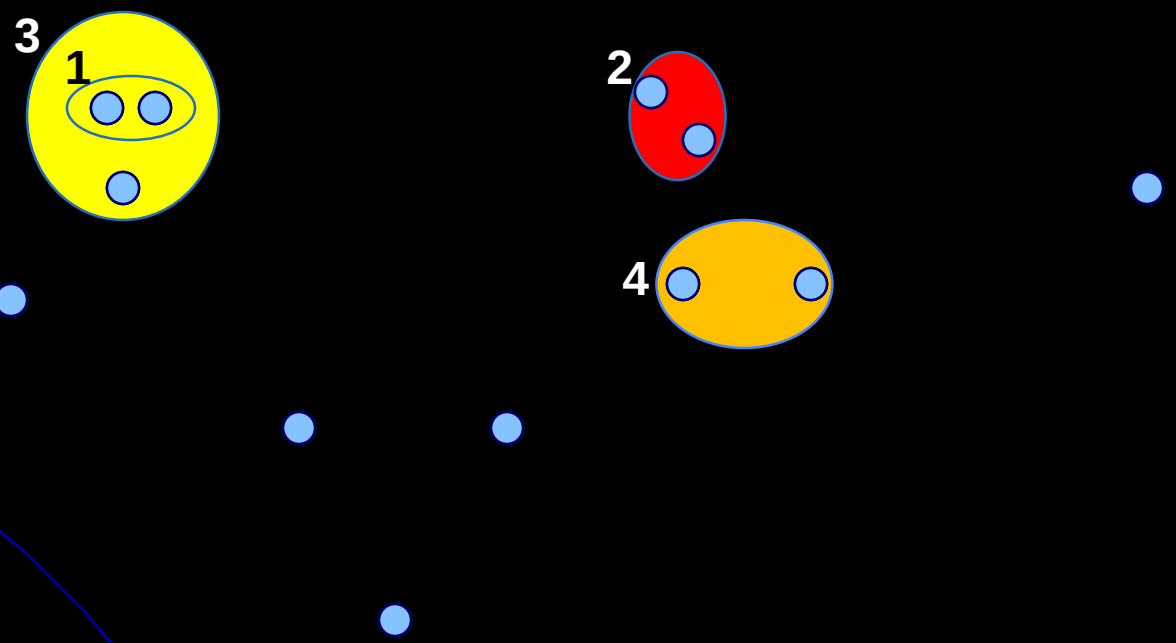
# Agglomerative clustering

3rd iteration



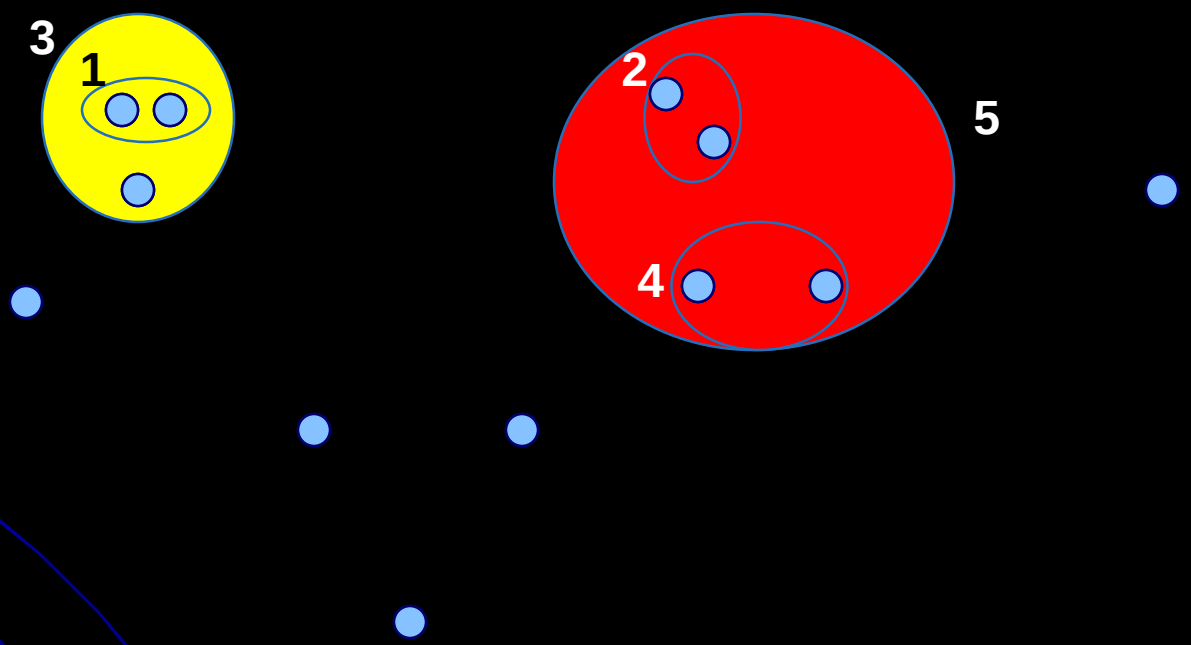
# Agglomerative clustering

4th iteration



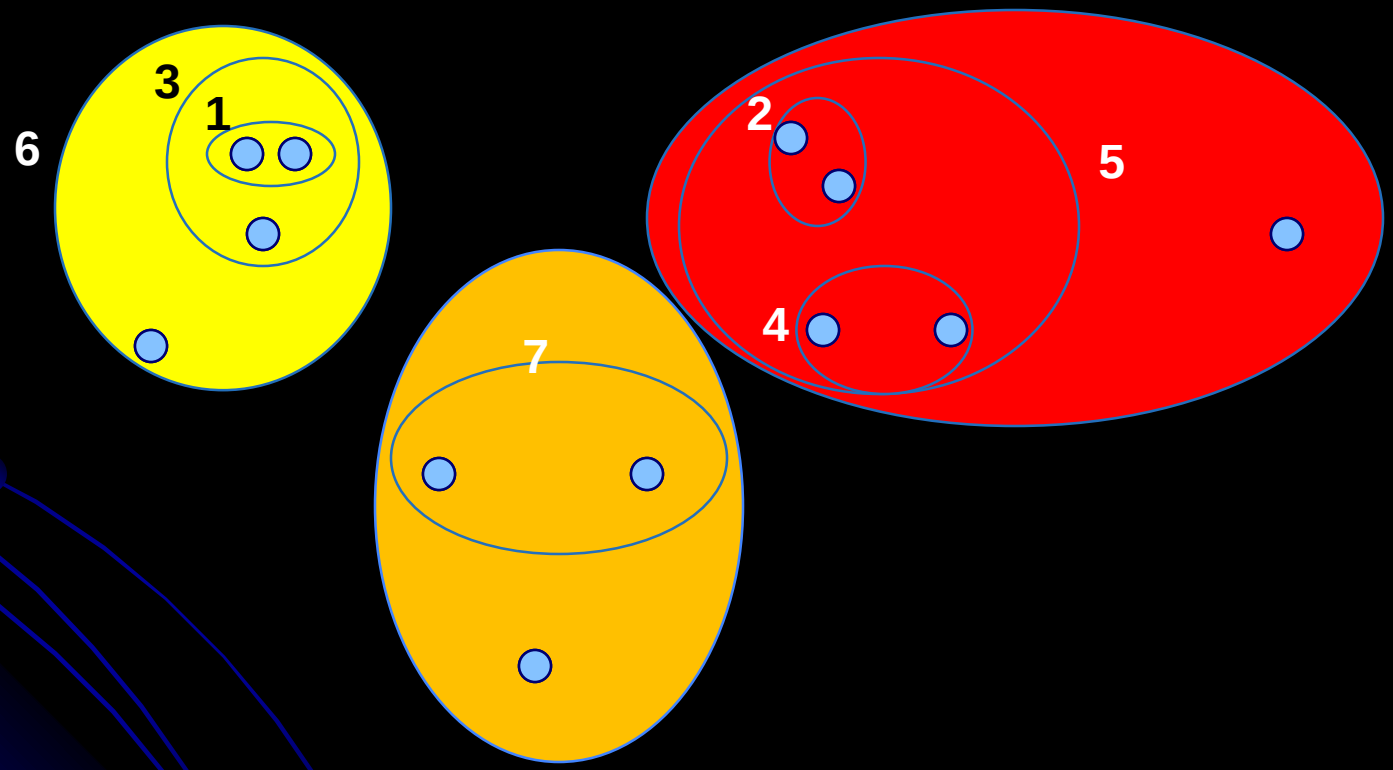
# Agglomerative clustering

5th iteration



# Agglomerative clustering

k clusters left



# Cluster distance

single-link:

Distance of the closest points from A and B

complete-link:

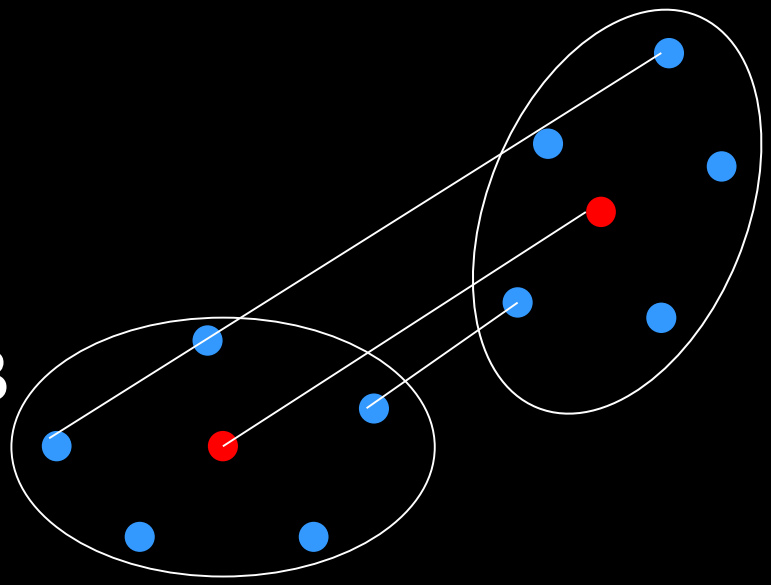
Distance of the furthest points from A and B

centroid-link:

Distance of the centroids

average-link:

Average distance between pairs of points from A and B



# Ward's method

Merge clusters with minimal merge cost:

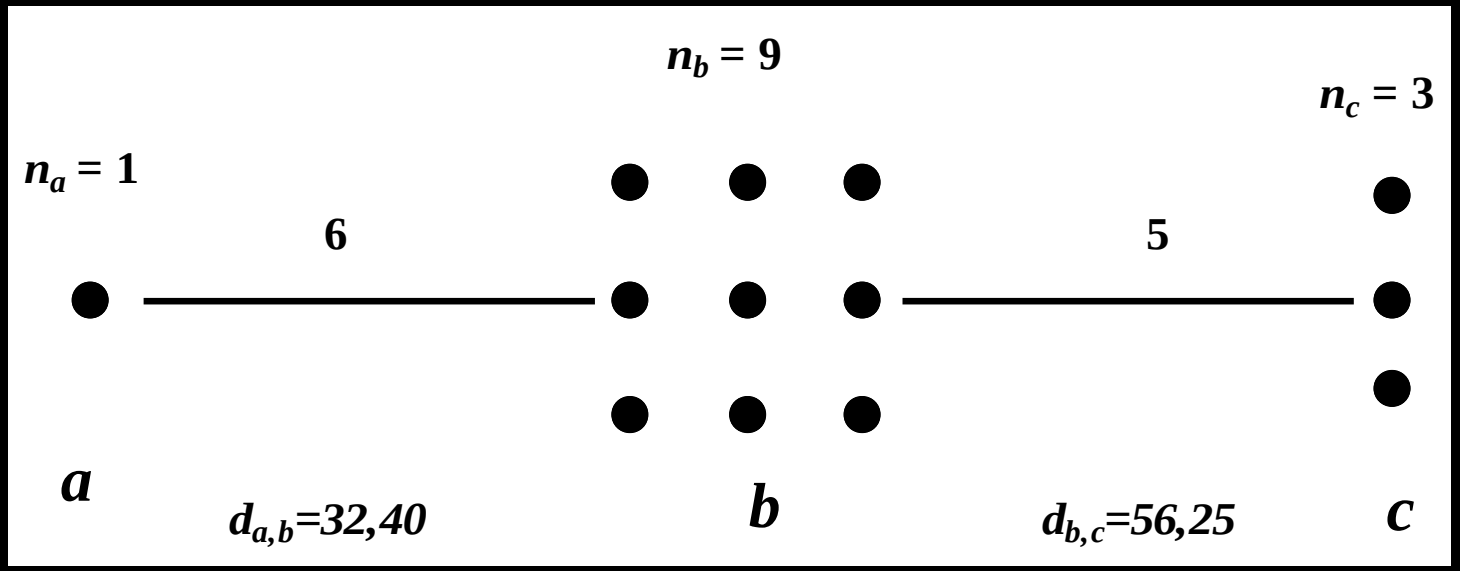
$$C_{A,B} = \frac{N_A N_B}{N_A + N_B} \|c_A - c_B\|^2$$

$N_k$ number of objects in cluster *K*

$c_k$ cluster centroid

J.H. Ward (1963): Hierarchical grouping to optimize an objective function, J. Am. Statist. Assoc. 58

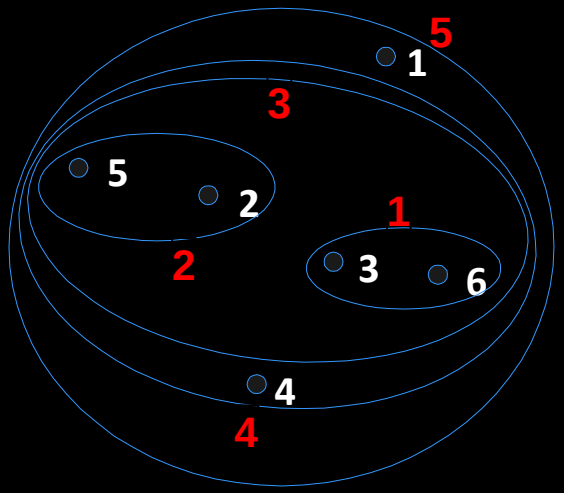
# Example of cost calculations




$$MergeCost(a,b) = \frac{1 \cdot 9}{1+9} \cdot 36 = \frac{9}{10} \cdot 36 = 32.40$$

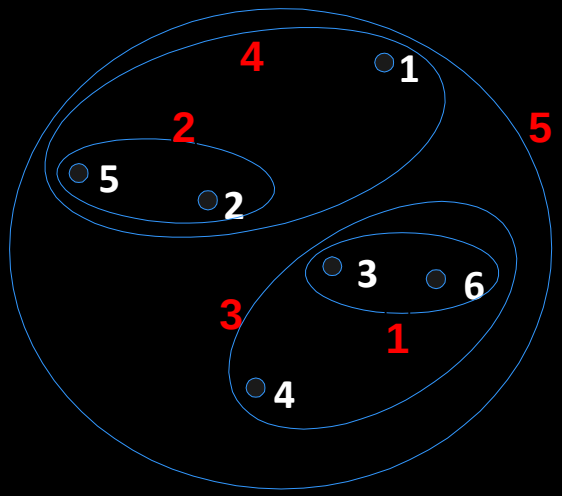
$$MergeCost(b,c) = \frac{3 \cdot 9}{3+9} \cdot 25 = \frac{27}{12} \cdot 25 = 56.25$$

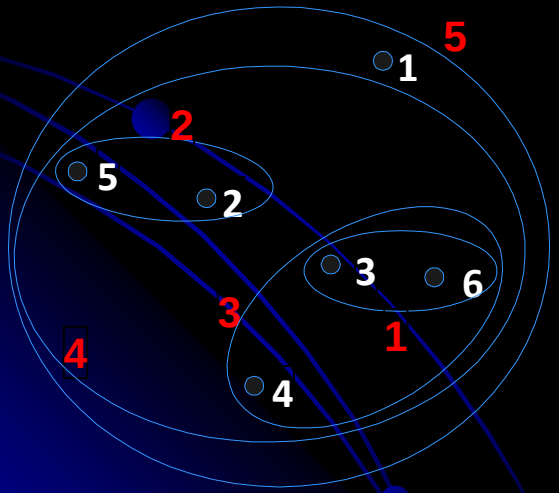
# Hierarchical Clustering: Comparison




MIN

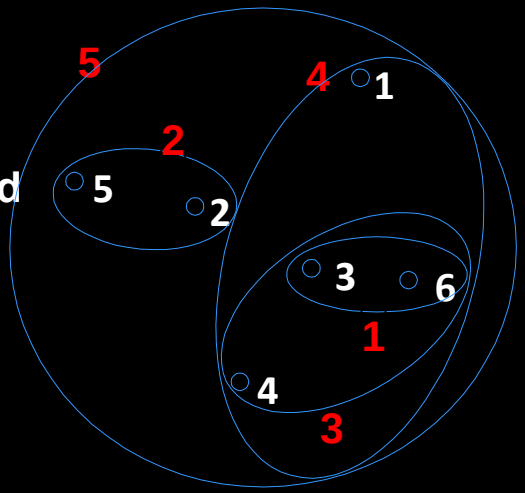
MAX




Group Average

Ward's Method



# Iterative shrinking

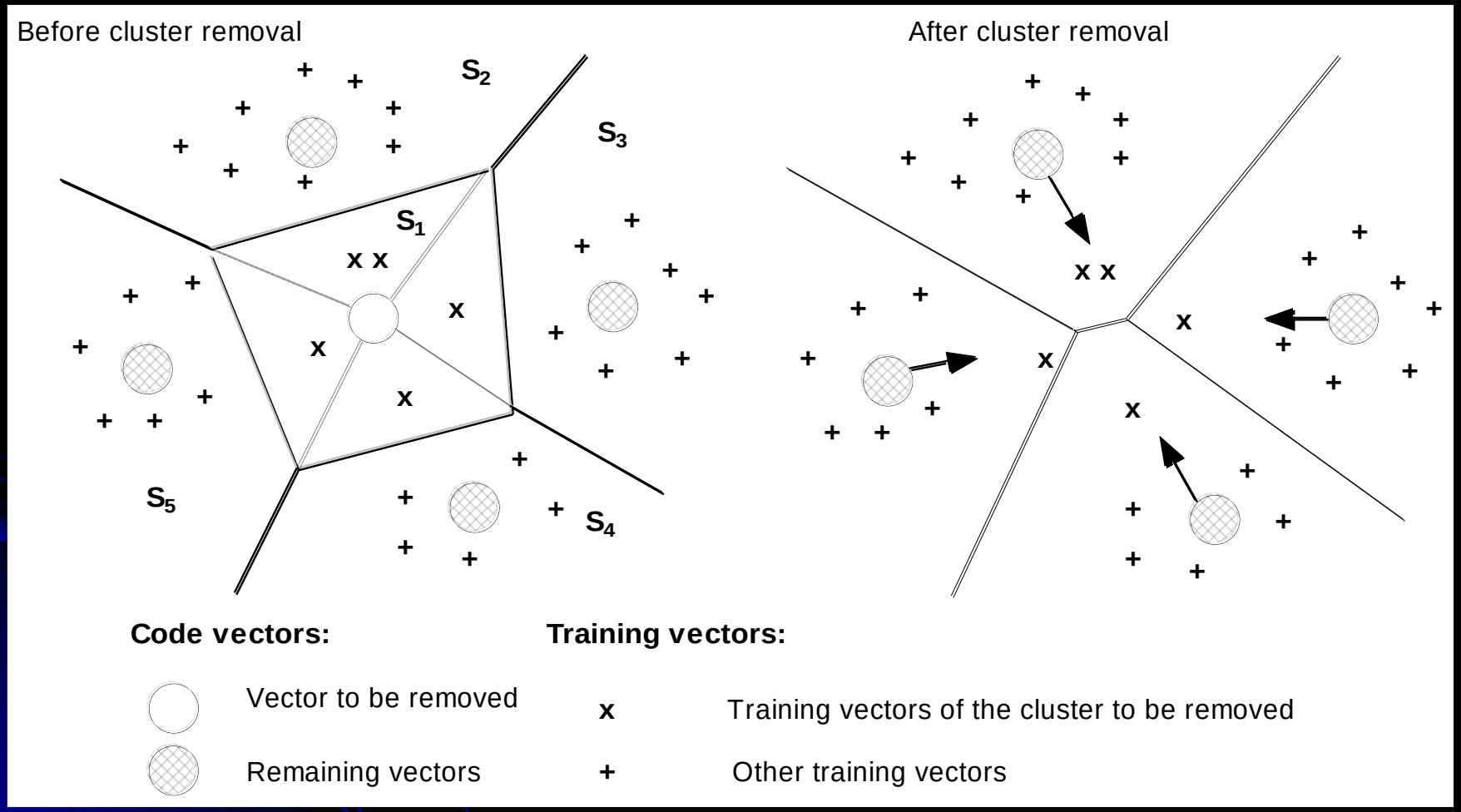
Individual objects form clusters
REPEAT
    Select cluster to remove (smallest removal cost)
    Repartition the object to the nearby clusters
UNTIL 1 cluster containing all objects
or desired number of clusters

$$C_A = \sum_{x_i \in A} \left( \frac{N_{Q_i}}{N_{Q_i} + 1} d(x_i, c_{Q_i}) - d(x_i, c_A) \right)$$

$Q_i$ nearest cluster for object $x_i$ having the minimal merge cost

Pasi Fränti, Olli Virmajoki (2006): Iterative shrinking method for clustering problems, Pattern Recognition, Volume 39

# Shrinking


Before cluster removal
After cluster removal
$S_1$
$S_2$
$S_3$
$S_4$
$S_5$
Code vectors:
Vector to be removed
Remaining vectors
Training vectors:
x
Training vectors of the cluster to be removed
+
Other training vectors